mikroKLAN 14

P. 1877/85

2 1985

informatyka

Język programowania BCPL

Emulator E6RM-E

Projektowanie baz danych

WYDAWNICTWO MON

Cena 100 zł

informatyka

2 1985

Nr 2

Miesięcznik Rok XX

Luty 1985

Organ Komitetu Informatyki MNSZWiT oraz Komitetu Naukowo-Technicznego NOT ds. Informatyki

KOLEGIUM REDAKCYJNE:

Mgr inż. Zbigniew GLUZA, dr inż. Wacław ISZKOWSKI, mgr Teresa JABŁOŃSKA (sekretarz redakcji), Władysław KLEPACZ (zastępca redaktora naczelnego), prof. dr hab. Leon ŁUKASZEWICZ (redaktor naczelny), mgr inż. Andrzej J. PIOTROWSKI, dr inż. Janusz ZALEWSKI

STALE WSPÓŁPRACUJĄ:

Mgr inż. Witold ABRAMOWICZ (Szwajcaria), mgr inż. Ryszard K. KOTT (Wielka Brytania), dr Jacek OWCZARCZYK, dr Andrzej SZAŁAS, dr Jakub TATARKIEWICZ, mgr inż. Teresa WILCZEK

PRZEWODNICZĄCY RADY PROGRAMOWEJ:

Prof. dr hab. Tadeusz PECHE

Materiałów nie zamówionych redakcja nie zwraca

Redakcja: 00-041 Warszawa, ul. Jasna 14/16, pok. 243 i 244, tel. 27-71-40 lub 26-82-61 w. 184

Zakł. Graf. „Tamka". Zam. 0057/85 Obj. 4,0 ark. druk. Nakład 6500 egz. N-10

ISSN 0542-9951, INDEKS 36124

Cena egzemplarza 100 zł
Prenumerata roczna 1200 zł

WYDAWNICTWO NACZELNA ORGANIZACJA TECHNICZNA
SIGMA
CZASOPISM I KSIĄŻEK TECHNICZNYCH
00-950 Warszawa
skrytka pocztowa 1004
ul. Biała 4

W NUMERZE: Strona

Język programowania BCPL (1) 1
Konrad Jabłoński

Emulator E6RM-E — przykład wykorzystania komputerów mikroprogramowanych 4
Marek Dziedzic, Krzysztof Perycz, Jerzy Wiliński

Projektowanie sieciowych baz danych na przykładzie SZBD-1100 6
Waldemar Kapuścik

Z KRAJU 10

Komputery w „N" resorcie
Gra finansowa na MERZE 400
Obsługa techniczna sprzętu w zestawieniu ankietowym

mikroKLAN 13

APPLE IIc
Rozszerzenie możliwości graficznych ZX81 (16K)
Współpraca z napędem dysków elastycznych PLX45D (2)
DS-80 MONITOR
Komputery z drugiej ręki
Akademia mikroKLANU (3)

ZE ŚWIATA 22

Najnowsze tendencje w dziedzinie komputerowych systemów sterowania
Rodzaje drukarek
NEW GENERATION COMPUTING

TERMINOLOGIA 30

Słownik pojęć i terminów z dziedziny grafiki komputerowej

CZWARTA OKŁADKA — *Rafał Pietrak, Jakub Tatarkiewicz*

W NAJBLIŻSZYCH NUMERACH:

- Wacław Iszkowski o mechanizmach komunikacji
- Andrzej Litwiniuk o kompilatorze LOGLANU dla MERY 400
- Zbigniew Kierzkowski i Jacek Małuszyński o współbieżnej aktualizacji bazy danych
- Julian Winiewski o języku C
- Frank Land o systemach informacyjnych
- Wojciech Cellary i Jan Węglarz o nauczaniu mikroinformatyki
- Wacław Iszkowski o superkomputerze CRAY
- Jan Bielecki o standardzie operacji wejścia-wyjścia w ADZIE
- Jerzy Karczmarczuk o efektywności obliczeń numerycznych na komputerach SINCLAIRA
- Halina Kontkiewicz-Chachulska o SICOBIE'84

P. 1877/85

W naszym cyklu prezentacji języków programowania i systemów operacyjnych (przypomnijmy, że począwszy od końca 1981 roku przedstawiliśmy w nim kolejno: ADĘ, LOGLAN, CP/M, BASIC, języki symulacyjne, IRT-FORTRAN, MODULĘ-2, PROLOG, FORTH, PL/M, CHILL i PL/Z) następuje dość ważny moment. Poniższym artykułem nawiązujemy do jednego z najciekawszych a jednocześnie trudno wytłumaczalnych zjawisk w historii programowania komputerów — sukcesu języka C i związanego z nim systemu operacyjnego UNIX.

O znaczeniu tego oprogramowania, znanego co najmniej od dziesięciu lat, przesądzono stosunkowo niedawno. Twórcy języka C, Brian Kernighan i Dennis Ritchie, otrzymali w roku 1983 nagrodę czasopisma ELECTRONICS, a rok później Dennis Ritchie i Ken Thompson zostali nagrodzeni bardziej liczącą się nagrodą im. Turinga, przyznawaną przez amerykańskie stowarzyszenie Association for Computing Machinery za wybitne osiągnięcia w dziedzinie techniki komputerowej.

Język C, w którym napisano prawie całość UNIXA i jego licznych odmian sięga korzeniami kilku innych języków, z których obecnie szerzej używany jest tylko jego bezpośredni poprzednik BCPL. Ze względu na znaczny wpływ języka BCPL na C w zakresie budowy i niektórych konstrukcji, prezentację tę rozpoczynamy właśnie od niego. Kompilatory obydwu języków stanowią narzędzia systemu UNIX, który omówimy w następnej kolejności (mniej więcej, od połowy roku). Zaraz po szerszym przedstawieniu UNIXA, a szczególnie jego jądra i tzw. powłoki (ang. shell), stanowiącej interpreter poleceń, których nazwy odpowiadają programom systemowym, przejdziemy do krótkiego omówienia innego ważnego narzędzia związanego z tym systemem, przeznaczonego do wytwarzania oprogramowania. Jest nim RATFOR, czyli RATIONAL FORTRAN, szerzej nie znany w Polsce preprocesor FORTRANU.

Prezentując Czytelnikom tę serię artykułów sądzimy, że nie trzeba specjalnie namawiać do ich lektury. Popularność UNIXA i jego narzędzi wzrasta tak gwałtownie, że z pewnością już niedługo, wraz z ekspansją sprzętu mikrokomputerowego, gorączka obejmie także Polskę.

Redakcja

KONRAD JABŁOŃSKI
ŚCO Cyfronet
Instytut Energii Atomowej
Świerk

Język programowania BCPL (I)

BCPL (ang. Basic Combined Programming Language), język programowania przeznaczony głównie dla minikomputerów, jest w Polsce praktycznie nieznany. W Instytucie Maszyn Matematycznych w 1975 roku dokonano próby wykorzystania kompilatora języka BCPL, działającego na komputerze IBM-370 — w celu generowania programów dla minikomputera MERA 400 [4]. W latach 1977—1978 dokonałem implementacji języka BCPL dla minikomputerów TPAi 1000 (prod. węgierskiej) i MERA 400 w Instytucie Badań Jądrowych. Oprócz wymienionego kompilatora BCPL dla komputera IBM-370, istnieje praktycznie nie używany kompilator na maszynie CYBER-73 w Świerku, oraz — będący w moim posiadaniu — kompilator dla minikomputera PDP-11, pracującego pod nadzorem systemu operacyjnego RSX-11D.

Mgr inż. KONRAD JABŁOŃSKI ukończył studia w 1964 r. na Politechnice Warszawskiej. Pracuje w ŚCO Cyfronet w Świerku, gdzie zajmuje się problematyką mikro- i minikomputerów, jak również systemami CAMAC i MULTIBUS II. W 1976 roku odbył roczny staż naukowy w KFA Julich w RFN, specjalizując się w programowaniu mikrokomputerów.

BCPL wywodzi się z języka CPL (ang. Combined Programming Language), który z kolei jest w znacznym stopniu [1] oparty na ALGOLU 60. Jego rodowód jest więc szlachetny i być może to właśnie spowodowało, że — podobnie jak ALGOL — nie zdobył sobie wystarczającej popularności. Z kolei z BCPL-u wywodzą się późniejsze języki, do których należy choćby język C, który w niektórych ośrodkach w Polsce jest już stosowany.

Z faktu, że protoplastą BCPL-u jest ALGOL 60 nie należy wyciągać wniosku, że jest to jedyna przyczyna braku popularności języka. Przede wszystkim język ten, podobnie jak wszystkie pozostałe, do pewnych celów nadaje się lepiej, do innych gorzej. Warto więc poznać obszar jego optymalnych zastosowań i tam go właśnie preferować. Skądinąd wiadomo, że w praktyce komputerowej stosuje się z różnych przyczyn nie te języki, które najlepiej nadają się do rozwiązywania problemów, a te, których znajomość w danym środowisku jest akurat najlepsza, bądź są dostępne na komputerach będących w zasięgu programistów. W dużym stopniu zależy to także od ich przyzwyczajenia.

Jaka jest więc dziedzina stosowania języka BCPL? Jest ona określona dość szeroko jako dziedzina zastosowań nienumerycznych, a więc dotyczy pisania różnego rodzaju programów systemów operacyjnych przetwarzających dane nienumeryczne bądź dane będące tylko liczbami naturalnymi, pisania kompilatorów, interpreterów, symulatorów, procesorów i edytorów tekstowych, programów obsługi (ang. handler), urządzeń we-wy itp. BCPL jako język wysokiego poziomu, zupełnie niezależny od typu maszyny, nadaje się więc także do komunikacji między ludźmi, np. przy stawianiu problemów do rozwiązania, bądź opisywaniu rozwiązywanych problemów w sposób skondensowany. Stosowany bywa na kursach akademickich jako narzędzie pomocnicze [6].

Właściwość BCPL pozwalająca stosować syntaktykę wywodzącą się z ALGOLU, wraz z możliwością operowania na najniższym poziomie maszyny (tzn. na jej rejestrach pa-

mięci), czynią go rodzajem języka superasemblera. Tak więc język BCPL najlepiej nadaje się do użycia tam, gdzie najskuteczniej problem rozwiązuje się za pomocą języków asemblera, przy czym w BCPL-u rozwiązuje się go o wiele szybciej i od razu dla dowolnej liczby komputerów różnej konstrukcji (!).

CHARAKTERYSTYKA JĘZYKA

Niewątpliwie każdy z języków sztucznie tworzonych, a do takich należą języki programowania komputerów, można scharakteryzować najściślej za pomocą opisu składni języka oraz elementów danych wykorzystywanych w języku. Taki opis BCPL można znaleźć w źródłowej pracy M. Richardsa [7]. Ze względu jednak na to, że praca ta jest w Polsce bardzo trudno dostępna, w celu umożliwienia Czytelnikom wstępnego przynajmniej zapoznania się z językiem, podaję niektóre jego właściwości składniowe, a także — opis podstawowego elementu danych.

Element BCPL

Jest to pojęcie utworzone na potrzeby języka, najbardziej zbliżone do pojęcia słowa maszynowego używanego w odniesieniu do rzeczywistych komputerów programowanych w językach asemblera. Każdy element BCPL składa się z dwóch części, nazwanych:

R — wartość elementu (ang. R-value)
L — wartość elementu (ang. L-value).

Wirtualna maszyna programowana w języku BCPL ma pamięć składającą się z komórek, pozwalających zapisać w nich wartości w postaci liczb naturalnych bądź odczytywać z nich te wartości. Każda komórka pamięci wirtualnej ma swój adres, który także można wyrazić za pomocą liczby naturalnej. Elementem BCPL jest więc połączenie adresu komórki pamięci maszyny wirtualnej (L-wartość) z zawartością tej komórki (R-wartość). Podobnie jak w maszynach rzeczywistych, tak i w opisywanej maszynie wirtualnej tylko R-wartość elementu może być modyfikowana, natomiast L-wartość określonego elementu nie może ulec zmianie. Jeśli wykonuje się operacje na L-wartości, to znaczy, że wyznacza się dopiero określony element BCPL.

Operacje na elementach

Do najważniejszych operacji należy przypisanie, co w BCPL-u zapisuje się jako:

```
VAR1:=VAR2
```

Oznacza to: R-wartość elementu nazwanego VAR1 zostaje zastąpiona przez R-wartość elementu nazwanego VAR2, przy czym po wykonaniu przypisania R-wartość elementu VAR2 nie ulega zmianie.

Oprócz zwykłej operacji przypisania, może wystąpić operacja przypisania R-wartości wyrażenia uzyskanej przy użyciu operacji artymetycznych, logicznych i specjalnych (typowych dla BCPL), zwanych operacjami adresowymi, na R-wartościach różnych elementów oraz wartościach stałych, np.:

```
VAR1:=VAR2+VAR1—37
```

Wartości stałe mogą być zapisywane w postaci nazw bądź liczbowo (głównie dziesiętnie i ósemkowo). Oprócz operatorów dodawania i odejmowania, można stosować operatory mnożenia i dzielenia, operatory sumowania i mnożenia logicznego, różnicy symetrycznej i jej negacji, a także negacji logicznej. Można też stosować następujące jedno- i dwuelementowe operatory adresowania, zdefiniowane na potrzeby języka BCPL: RV, LV i %.

Instrukcja przepisania:

```
VAR1:=RV VAR2
```

oznacza, że R-wartość elementu VAR2 jest traktowana jako L-wartość pewnego elementu X, którego R-wartość zostaje przypisana R-wartości elementu VAR1. Tak więc jest to przypisanie z wykorzystaniem adresowania pośredniego przy odczycie. Liczba operatorów RV może być dowolnie duża, przez co można uzyskać dowolnie głęboko sięgające adresowanie pośrednie odczytu.

Działanie operatora LV, jak w przykładzie:

```
VAR1:=LV VAR2
```

tłumaczy się w ten sposób, że R-wartości elementu VAR1 zostaje przypisana L-wartość elementu VAR2. Nie ma więc sensu dwukrotne użycie operatora LV.

W instrukcji przypisania niektóre operatory adresowania mogą być użyte także i po lewej stronie, np.:

```
RV VAR1:=VAR2
```

co oznacza, że R-wartość elementu VAR2 jest przypisana R-wartości pewnego elementu, którego L-wartość jest R-wartością elementu VAR1. Jest to więc pośrednie adresowanie przy zapisie!

Umieszczenie operatora LV po lewej stronie znaku przypisania jest niedopuszczalne, gdyż oznaczałoby zmianę adresu komórki pamięci w maszynie wirtualnej, co z założenia jest niemożliwe.

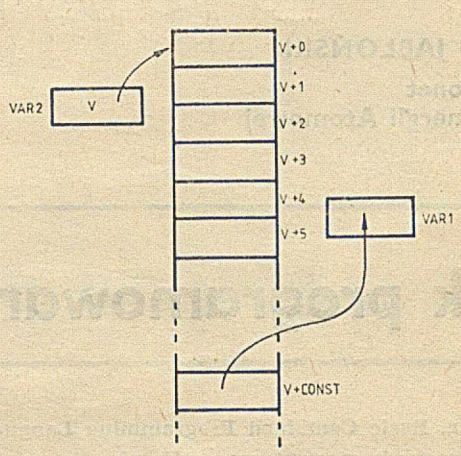
Oprócz jednoelementowych operatorów adresowania stosuje się operator dwuelementowy (przemienny):

```
VAR1:=VAR2 % CONST
```

lub

```
VAR1:=CONST % VAR2
```

Jeden z argumentów prawej strony przypisania musi być elementem, natomiast drugi — wartością stałą. Wartość wyrażenia po prawej stronie jest R-wartością elementu o L-wartości uzyskanej z dodania R-wartości elementu VAR2 i stałej CONST, co ilustrowano na rysunku. Wyrażenie VAR % CONST można zapisać jako RV (VAR+CONST).



Ilustracja operacji VAR1:=VAR2%CONST

W BCPL używa się elementu zwanego wektorem dla oznaczania liniowo uporządkowanych zbiorów liczb (ang. array). R-wartość takiego elementu jest adresem zerowego elementu zbioru, a każdy następny element zbioru zajmuje komórkę pamięci wirtualnej o adresie o 1 większym niż element poprzedni. Pozwala to na arytmetyczne obliczanie adresu elementów wektora, np. w instrukcji FOR:

```
FOR I=1 TO GETBYTE(S,0) DO WRCH (GETBYTE(S,I))
```

W tym przypadku, po zwiększeniu o 1 wskaźnika I, pobierane są kolejno R-wartości elementów zbioru S, przy czym zerowy element zawiera liczbę elementów zbioru (jest to zalecenie języka BCPL).

Deklaracje

Wszystkie zmienne bez wyjątku muszą być deklarowane, natomiast stałe muszą być wtedy deklarowane gdy są wyrażone przez nazwę. W BCPL rozróżnia się cztery rodzaje deklaracji.

Deklaracje MANIFEST dotyczą stałych używanych w programie. Ich wartości muszą być ustalone już w fazie kompilacji programu i mogą być deklarowane w postaci wyrażeń, których wszystkie składniki muszą być znane w chwili kompilacji, np.:

```
MANIFEST ($
            FILE=0
            ENTRY.WORD.3=#77
            EOF=25
          $)
```

Deklaracje STATIC dotyczą zmiennych definiowanych w obszarze pamięci wirtualnej nie używanej ani na stos, ani na instrukcje programu. Zmienne te odpowiadają komórkom roboczym w pamięci operacyjnej komputera programowanego w języku asemblera. Zwykle nadaje się im obojętną wartość początkową, najczęściej 0. Znaczy to, że w chwili wprowadzenia programu do rzeczywistego komputera tak zadeklarowane zmienne będą miały R-wartości według deklaracji, np.

```
STATIC ($
        FLAG=0
        MAIN.FLAG=TRUE
       $)
```

Program pisany w BCPL nie musi być kompilowany w całości, co w praktyce daje rozliczne korzyści. Jednakże, w celu zapewnienia łączności między poszczególnymi segmentami programu, a także — łączności z biblioteką, którą należy wtedy traktować jako wcześniej opracowany segment programu, wprowadza się w BCPL tzw. wektor komunikacyjny, który ma charakter globalny. Deklaracja GLOBAL oznacza więc przyporządkowanie deklarowanej nazwy określonej komórce wektora. Dostęp do tej komórki może odbywać się przez jedną nazwę lub różne nazwy, ale zadeklarowane dla tej samej komórki. Względne numery komórek wektora deklaracji GLOBAL muszą być liczbami znanymi w czasie kompilacji. Rozmiar wektora oraz przyporządkowanie nazw bibliotecznych poszczególnym komórkom wektora zależy od implementacji.

Deklaracja zmiennych LET stanowi czwartą grupę deklaracji i dotyczy zmiennych dynamicznych, tworzonych dopiero w czasie wykonywania programu na stosie maszyny wirtualnej. Zmienne deklarowane jako dynamiczne mogą w czasie wykonywania programu fizycznie występować wielokrotnie w różnych miejscach stosu, lecz dzięki ich lokalności nie powoduje to konfliktu. Także dzięki lokalności zmiennych dynamicznych, ta sama nazwa może być użyta do definiowania różnych zmiennych, pod warunkiem zachowania warunków lokalności.

Uzupełnieniem deklaracji zmiennych jest deklarowanie procedur i funkcji, co odbywa się także przy użyciu słowa LET. Oto formalne różnice w deklaracjach poszczególnych obiektów (podwójny znak łamania „//" oznacza komentarz):

```
LET A=TREEMAX //Deklaracja zmiennej dynamicznej
LET WRITECHARD() BE...//Deklaracja procedury
LET FORMTREE=VALOF...//Deklaracja funkcji
```

Instrukcje

W BCPL, podobnie jak w innych językach programowania instrukcje służą do opisu pewnych czynności na elementach danych, są poleceniami wykonania tych czynności. Na etapie kompilacji programu instrukcje stanowią jedynie schematy działań, które będą wykonywane dopiero w czasie wykonywania programu. Z konieczności omówione mogą być tu jedynie wybrane instrukcje języka.

Do podstawowych instrukcji w BCPL należy instrukcja przypisania oznaczana symbolicznie znakiem := , której działanie wyjaśniono przy omawianiu elementu BCPL. Przy użyciu instrukcji przypisania tworzy się inne, bardziej złożone instrukcje, które dokonują pewnych modyfikacji elementów lub służą do kontroli programu. W BCPL szereg występujących po sobie instrukcji przypisania można zastąpić jedną, zawierającą po lewej i po prawej stronie listy wyrażeń. Powoduje to oczywiste skrócenie zapisu; np. zamiast:

```
A :=A+1
B :=0
C :=FALSE
```

można napisać:

```
A, B, C :=A+1, 0, FALSE
```

Wywołanie procedury w BCPL jest instrukcją, którą zapisuje się jako nazwę zakończoną nawiasami okrągłymi, zawierającymi parametry wywoływanej procedury. W szczególnym przypadku procedura może nie mieć żadnych parametrów. W przypadku wywołania procedury:

```
WRITE.CHARACTER („G")
```

parametr jest znakiem alfanumerycznym (literą G). Bezparametrowe wywołanie procedury może mieć postać:

```
NEWLINE ()
```

gdyż w tym przypadku polecenie dotyczy z góry zdefiniowanych czynności.

Bardzo bogatą grupę instrukcji stanowią w BCPL, instrukcje warunkowe do których należą — na przykład — UNTIL i REPEATUNTIL. Obie wymienione instrukcje wymagają badania wyrażenia, którego wartość logiczna rozstrzyga o dalszym postępowaniu. Przykładowo zapis:

```
UNTIL B=6
    B :=B+1
```

oznacza, że tak długo dopóki R-wartość elementu B nie będzie równa 6 (wyrażenie logiczne) następuje dodawanie jedynek do R-wartości elementu B. W szczególnym przypadku, gdy przed wykonaniem instrukcji UNTIL R-wartość B była równa 6, instrukcja przypisania nie zostanie w ogóle wykonana.

W przypadku instrukcji:

```
B :=B+1
    REPEATUNTIL B=6
```

najpierw nastąpi bezwarunkowe dodanie jedynki do R-wartości elementu B, a dopiero potem — zbadanie wyrażenia logicznego, którego wynik rozstrzygnie o powtórnym dodaniu jedynki. Przy tak zapisanej instrukcji może zdarzyć się, że dla początkowej R-wartości elementu B równej 6 instrukcja będzie wykonywana w nieskończonej pętli, zgodnie z nakazem powtarzania dodawania jedynek.

Przykładem typowej instrukcji sterującej jest instrukcja RETURN, która nakazuje przerwać wykonywaną procedurę i powrócić do programu wywołującego tę procedurę.

LITERATURA

[1] Barron D. W., Buxton J. N., Hartley D. F., Nixon E., Strachey C.: The Main Features of CPL, Computer Journal, vol. 6 (1963), pp. 134—143

[2] Holden J.: BCPL/RSX-11D User Manual. Department of Computer Science, University of York, Heslington, June 1976

[3] Jabłoński K.: BCPL — język superasemblera. Postępy Techniki Jądrowej, vol. 24, str. 383—393 (1980)

[4] Kompilator skrośny języka BCPL maszyny cyfrowej MERA 400, Instytut Maszyn Matematycznych, Warszawa, 1975

[5] Richards M.: A Description of the BCPL Compiler Phase 1: BCPL to OCODE. University of Cambridge, Computer Laboratory, August 1971

[6] Richards M., Whitby-Stevens C.: BCPL — The Language and its Compiler. Cambridge University Press, 1979

[7] Richards M.: The BCPL Programming Manual. University of Cambridge, Computer Laboratory, April 1973

[8] Suffrin B. A.: A BCPL Implementation. University of Essex.

Wyniki konkursu

Zarząd Główny i Zarząd Sekcji Cybernetyki Medycznej Internistów Polskich ogłosił w 1983 r. konkurs pn. **„Algorytm i program komputerowy rekonstrukcji obrazów tomograficznych metod CT i NMR** (p. INFORMATYKA nr 7—8, 1983 r.).

W styczniu br. na sympozjum na temat tomografii MR, zorganizowanym przez SC-TIP, IFM-PAN oraz PLTR w Katowicach nastąpiło jego rozstrzygnięcie. Krótkie omówienie algorytmów i programów znajduje się w tomie streszczeń referatów prezentowanych na tym sympozjum.

A oto wyniki konkursu i przyznane nagrody:

I nagroda w wysokości 50 000 zł dla zespołu: P. Penczek, P. Franaszczuk, K. Leszczyński z Pracowni Fizyki Medycznej Instytutu Fizyki Doświadczalnej Uniwersytetu Warszawskiego

II nagroda w wysokości 20 000 zł — dla W. Malinowskiego z Instytutu Fizyki Molekularnej PAN w Poznaniu

Dwie równorzędne II nagrody po 20 000 zł
- — dla zespołu: Z. Marszałek, J. Czopik, K. Nowak, A. Trojnar z Politechniki Śląskiej i Hutniczego Przedsiębiorstwa Remontowego
- — dla A. Połacina z Zakładu Elektroniki Jądrowej i Medycznej Instytutu Radioelektroniki Politechniki Warszawskiej

III nagroda w wysokości 10 000 zł — dla Z. Drużyckiego występującego jako osoba prywatna

nagroda dodatkowa w wysokości 5000 zł — dla J. Czaplickiego z Instytutu Fizyki Molekularnej PAN w Poznaniu.

MAREK DZIEDZIC
KRZYSZTOF PERYCZ
JERZY WILIŃSKI
ZETO Gdańsk

Emulator E6RM-E — przykład wykorzystania komputerów mikroprogramowanych

Wprowadzenie komputerów Jednolitego Systemu oraz planowane wygaśnięcie produkcji maszyn cyfrowych serii ODRA narzuciły konieczność wykorzystania dotychczasowego bogatego oprogramowania systemu ODRA w środowisku komputerów Jednolitego Systemu (RIAD). Temat ten przewijał się w wielu już publikacjach, m.in. [1, 2, 3, 8, 10].

Metody przenoszenia oprogramowania

Istnieją dwie zasadnicze metody przenoszenia oprogramowania na inną maszynę cyfrową [8, 10]:

- **metoda przeprogramowania,** w której stosuje się różne sposoby translacji programu na poziomie tekstu źródłowego, połączone z ewentualną konwersją zbiorów danych
- **metoda modelowania,** która umożliwia bezpośrednie, interpretatywne wykonywanie programu binarnego komputera macierzystego na nowym komputerze; metoda ta, jeśli jest zrealizowana wyłącznie przy użyciu środków programowych nowego komputera, nazywa się **symulacją,** natomiast gdy część lub całość procesu modelowania jest realizowana sprzętowo, nosi nazwę **emulacji.**

Pierwsza z wymienionych metod ma ograniczone zastosowanie, ponieważ nie zapewnia ani dużej efektywności, ani całkowitej zgodności programowej — z uwagi na nieuwzględnianie dynamicznej struktury programu na etapie jego translacji. Trudności napotykane przy stosowaniu tej metody podaje szerzej opracowanie [8].

Druga z wymienionych metod zapewnia uzyskanie pełnej zgodności programowej bez konieczności ingerencji w istniejące oprogramowanie. Ogólną dyskusję zalet i wad symulacji i emulacji zawiera opracowanie [8], zaś konkretne realizacje symulatorów komputera ODRA 1305 na R-32 omówione są w pracach [2, 3], a emulatora R6RM-E — w opracowaniu [1].

Wybór technicznej metody realizacji emulatora komputera macierzystego na nowym komputerze jest w oczywisty sposób uwarunkowany jego strukturą.

Rodzaje maszyn cyfrowych

Można rozróżnić dwie podstawowe klasy maszyn cyfrowych: zawierające oraz nie zawierające mikroprogramowanych układów logicznych.

Procesor maszyn cyfrowych był tradycyjnie wykonywany jako sieć kombinacyjno-sekwencyjnych układów logicznych, której poszczególne części pełniły wyspecjalizowane funkcje zdefiniowane przyjętym repertuarem instrukcji maszynowych. O ile w przypadku mniejszych systemów komputerowych takie podejście konstrukcyjne było uzasadnione, to w większych maszynach, dysponujących szerszymi możliwościami przetwarzania, powodowało nadmierny wzrost złożoności sprzętu, a w konsekwencji — znaczny wzrost kosztów jego zaprojektowania i wytwarzania. Podejście to stosowane jest aktualnie tylko w nielicznych przypadkach, gdy krytycznym parametrem konstrukcyjnym komputera jest szybkość.

Stosowana obecnie technika mikroprogramowanych układów logicznych, ma — oprócz prostoty i uporządkowania procesu projektowania [11, 12] — wiele dodatkowych zalet, jak:

- **elastyczność:** zamiast zmian w układach procesora wystarczy zmodyfikować mikroprogram
- **ułatwienia dla emulacji:** przez zmianę mikroprogramów uzyskuje się inną interpretację instrukcji maszynowych
- **„mocniejsze" instrukcje:** możliwa jest realizacja skomplikowanych instrukcji, a nawet zdań języka wysokiego poziomu; istnieje także możliwość tworzenia specjalizowanych procedur użytkownika
- **konserwacja, diagnostyka i szkolenie:** prostota logiki mikroprogramowanej ułatwia wykrywanie i usuwanie błędów sprzętowych oraz upraszcza proces szkolenia personelu technicznego.

Obszerniejsze wiadomości na temat mikroprogramowania podają publikacje [4, 9, 13, 14].

W technice tej realizuje się układowo wybrany zestaw operacji elementarnych (mikrooperacji); każda z instrukcji maszynowych wymaga wykonania pewnej ich liczby. A które z nich i w jakiej kolejności mają zostać wykonane — tym steruje odpowiedni mikroprogram, umieszczony w pamięci mikroprogramów.

W skład jednego mikrorozkazu wchodzi wiele mikrooperacji, sterujących jednocześnie rozmaitymi równolegle realizowanymi funkcjami, oraz możliwy do modyfikowania adres następnego mikrorozkazu, co umożliwia tworzenie łańcuchowanych mikroprogramów zawierających pętle i rozgałęzienia. Możliwymi do jednoczesnego wykonywania są nie tylko mikrooperacje dotyczące oddzielnych bloków funkcjonalnych komputera (jednostki centralnej, pamięci operacyjnej, systemu wejścia-wyjścia), ale również w ramach tych bloków funkcjonalnych jest możliwa praca równoległa. Dzięki takiemu podejściu, uzyskuje się znaczne przyspieszenie działania systemu liczącego.

Wiele współczesnych komputerów stosuje rozbudowane, sterujące jednocześnie dużą liczbą mikrooperacji, mikrorozkazy o długości przekraczającej 100 bitów. Pamięć mikroprogramów jest realizowana wielorako: jako pamięć stała typu ROM, jako pamięć programowalna typu EPROM lub jako pamięć typu RAM, której zawartość wczytuje się ze specjalnego urządzenia pamięci zewnętrznej. Zawartość tego typu pamięci mikroprogramów może być czasem również dynamicznie zmieniana specjalnymi instrukcjami w czasie pracy komputera.

Struktura logiczna komputera R-32

Komputer R-32 należy do grupy maszyn mikroprogramowanych i składa się z następujących bloków funkcjonalnych:

- **procesora,** obejmującego arytmometr, pamięć lokalną oraz układy sterowania wraz ze stałą pamięcią mikroprogramów
- **pamięci operacyjnej**
- **kanałów wejścia-wyjścia,** umożliwiających wymianę informacji między pamięcią operacyjną a urządzeniami wejścia-wyjścia

koordynatora kanałów, umożliwiającego współpracę kanałów z procesorem.

W skład arytmometru wchodzą m.in. rejestry wewnętrzne, liczniki (w tym licznik rozkazów), sumator słowowy (32-bitowy) oraz sumator bajtowy, realizujący także operacje logiczne, takie jak: OR, AND i EXCLUSIVE OR, komutator procesora (sterujący przesyłaniem danych między rejestrami wewnętrznymi oraz wykonujący przesunięcia o jeden lub cztery bity). Pamięć lokalna zawiera m.in. dostępne programowo rejestry ogólne i zmiennoprzecinkowe.

W skład układów sterowania wchodzi m.in. układ synchronizacji, określający kolejność realizacji mikrooperacji, układy sterowania pamięcią mikroprogramów oraz pamięcią operacyjną i lokalną, wskaźniki i rejestry wewnętrzne (w tym słowo stanu programu — PSW).

Jak już wspomniano, układy logiczne jednostki centralnej realizują algorytmy związane z obsługą kanałów wejścia-wyjścia.

Jednostka centralna działa zatem w dwóch stanach: procesorowym bądź kanałowym, różniących się sposobem interpretacji niektórych pól mikrorozkazu, a także tym, że mikroprogram w stanie procesorowym może zostać przerywany przez żądanie obsługi kanału. W tym przypadku aktywowany jest odpowiedni mikroprogram działający w stanie kanałowym (w tym stanie dalsze przerwania są zablokowane), po wykonaniu którego następuje powrót do realizacji przerwanego mikroprogramu. Szczegółowy opis działania zawarty jest m.in. w dokumentacji [7].

Pamięć mikroprogramów zawiera 3072 słowa 86-bitowe. Jest ona zrealizowana jako pamięć stała transformatorowa. Na jedno słowo przypada 86 rdzeni. Adres słowa jest wyznaczony przez położenie punktów lutowniczych w matrycach — diodowej i tranzystorowej — między którymi przeciągnięty jest przewód pobudzający. Gdy przechodzi on przez rdzeń transformatora — generowana jest przy odczycie logiczna jedynka, a w przeciwnym wypadku — zero.

Dostępna przestrzeń adresowa obejmuje 4096 słów (rejestr adresowy pamięci mikroprogramów ma 12 bitów).

Mikroprogramy sterują wykonywaniem instrukcji maszynowych, pracą kanałów wejścia-wyjścia oraz realizują okresowe testy diagnostyczne. Komputer R-32 ma bogaty zestaw mikrooperacji, z których niektóre są wyspecjalizowane (np. mikrooperacja dekodująca instrukcje maszynowe), inne zaś są wystarczająco ogólne, by można było zrealizować szeroki zestaw algorytmów, w tym np. realizujących rozkazy z repertuaru maszyny ODRA (dokumentacja [5]).

Emulator E6RM-E

Jak wspomniano na wstępie — emulator, i symulator realizują wspólne zadanie, jakim jest modelowanie maszyny macierzystej na nowej maszynie, nieco odmiennymi środkami. Oba te narzędzia łączy jedność stosowanych algorytmów, dzieli jedynie sposób realizacji.

Wyróżnić można niejako dwa działy symulatora, które w różnym stopniu opłaca się wspomagać sprzętowo przechodząc do emulacji:

- **dział „organizacji obliczeń"**, czyli zamodelowany zespół funkcji wykonywanych normalnie przez system operacyjny (egzekutor), a więc realizacja wieloprogramowości, zarządzanie zasobami systemu komputerowego, wykonywanie ekstrakodów i komunikacja z operatorem
- **dział „obliczeń właściwych"**, czyli model procesora maszyny macierzystej, tzn. pętla pobierania i dekodowania instrukcji oraz algorytmy realizacji instrukcji nieekstrakodowych.

Bezdyskusyjna jest opłacalność sprzętowego wsparcia działu drugiego, natomiast sprzętowa realizacja funkcji działu pierwszego, ze względu na ich charakter i rozległy zakres — nie ma uzasadnienia.

Zasadniczą barierę ograniczającą pole zastosowań symulatora [2] stanowi szybkość przetwarzania. Według przybliżonych ocen, symulowany procesor ODRY wykonuje na komputerze R-32 ok. 10—15 tys. operacji/s. Naturalnym zatem celem rozwoju symulatora stał się emulator, wykorzystujący z jednej strony dobre i sprawdzone algorytmy symulatora, a z drugiej — istniejące możliwości mikroprogramowania komputera R-32. Podstawowe straty czasowe w symulatorze powoduje pętla pobierania i dekodowania instrukcji ODRY, dlatego też została ona „zamikroprogramowana" w pierwszej kolejności. Umieszczenie jej (metodą „szycia") w pamięci mikrorozkazów powinno spowodować około czterokrotny wzrost szybkości (do ok. 50 tys. operacji/s). Ze względu na ograniczoną pojemność zrealizowanej w komputerze R-32 pamięci mikroprogramów, nie przewiduje się na tym etapie dopisywania dodatkowych mikrorozkazów, wykonujących instrukcje maszynowe ODRY.

Przestrzeń adresowa pamięci mikroprogramów obejmuje 4096 słów. Ocenia się, że w nieużywanej jej części (1 K słów) istnieje możliwość umieszczenia mikroprogramów dla wszystkich realizowanych sprzętowo instrukcji ODRY. Jednak realizacja takiego emulatora wiąże się z koniecznością wymiany całej pamięci mikroprogramów. Przy obecnych krajowych możliwościach technologicznych nie jest to zadanie zbyt skomplikowane. 16 płyt obecnie stosowanej pamięci typu transformatorowego można zastąpić czterema płytami pamięci półprzewodnikowej typu PROM — zawierającej wszystkie mikroprogramy komputera R-32 oraz dodatkowo mikroprogramy emulujące maszynę ODRA. Spodziewana prędkość emulacji przy takim rozwiązaniu wzrośnie co najmniej do ok. 150 tys. operacji/s.

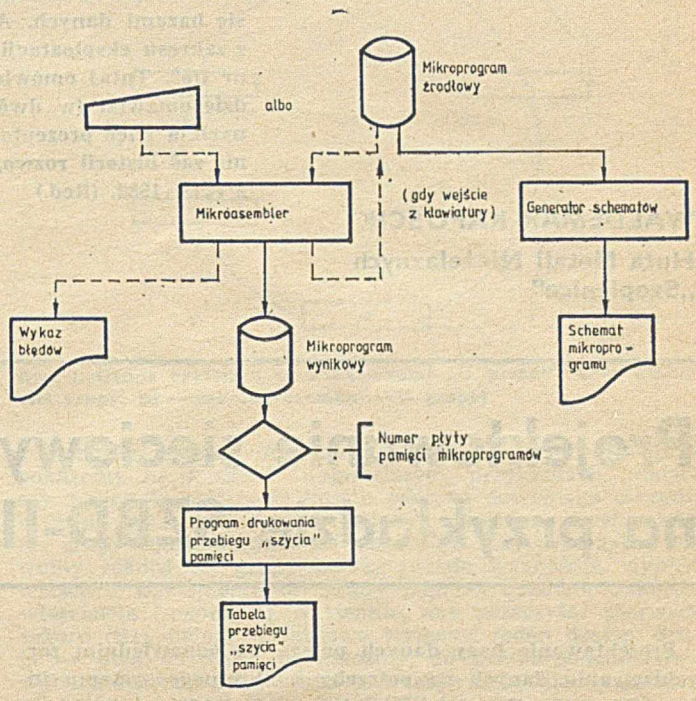
Programy wspomagające mikroprogramowanie

W celu ułatwienia pisania mikroprogramów, zmniejszenia możliwości pomyłek oraz zapewnienia automatycznego tworzenia dokumentacji, zostały napisane w ramach prac nad emulatorem E6RM-E, specjalne programy wspomagające, które działają na maszynie ODRA 1305 pod nadzorem systemu operacyjnego GEORGE-3.

Do pisania mikroprogramów opracowano język zbliżony do języka asemblera. W zapisie źródłowym podawane są tylko niezbędne mikrooperacje oraz adresy mikrorozkazów: bieżącego i następnego. Mikroprogram źródłowy jest weryfikowany i kompilowany za pośrednictwem mikroasemblera. Sprawdzony mikroprogram źródłowy stanowi dane wejściowe dla programu generującego schemat logiczny w formacie zbliżonym do używanego w istniejącej dokumentacji [5].

Mikroprogram wynikowy, po posortowaniu zgodnie z numerem płyty pamięci mikroprogramów komputera R-32, może być wydrukowany w formacie tabeli przebiegu „szycia" pamięci, analogicznej jak w dokumentacji [6].

Zależności między wspomnianymi programami obrazuje rysunek.



Schemat zależności pomiędzy programami

Możliwość emulacji komputera ODRA na innych modelach Jednolitego Systemu

Komputery Jednolitego Systemu są rodziną zgodną programowo. Poszczególne modele różnią się między sobą jedynie realizacją techniczną i mocą obliczeniową. Istnieją również modele zrealizowane bez użycia mikroprogramowania (np. R-50), na których wykonanie zaprezentowanego sposobu emulacji w ogóle nie jest możliwe (można stosować tylko symulację).

Istnieją również, bądź są zapowiedziane, modele z programowalną pamięcią mikroprogramów (modele 25, 35, 45, 55 i 60 oraz zapowiedziany w kraju — R-34). Realizacja emulatora na tych modelach oczywiście nie wymaga żadnej ingerencji w sprzęt.

Pozostałe komputery Jednolitego Systemu, w szczególności krajowy R-32, wymagają dla zrealizowania emulatora modyfikacji bądź całkowitej wymiany pamięci mikroprogramów.

Z uwagi na różnice w formatach mikrorozkazów różnych modeli Jednolitego Systemu, mikroprogramy emulacji będą się istotnie różniły, natomiast pozostała, programowa część emulatora, w zasadzie nie ulegnie zmianie.

LITERATURA

[1] Dziedzic M., Perycz K., Wiliński J.: Emulacja maszyn serii ODRA 1300 na komputerach RIAD-32. Mat. konferencyjny pt. „Perspektywy rozwojowe systemów cyfrowych na komputerze ODRA 1305". ZETO Wrocław, 1983

[2] Dziedzic M., Perycz K., Wiliński J.: Wieloprogramowy dyskowy symulator MC ODRA 1305 na MC RIAD-32. Biuletyn Informacyjny ZETO Gdańsk, Gdynia 1983

[3] Fręśko Z., Perycz K., Wiliński J.: Symulator — założenia i dokumentacja projektowo-programowa. Wyd. wewn. ZETO Gdańsk, Gdynia 1978

[4] Husson S. S.: Microprogramming: Principles and Practice. Prentice-Hall, Englewood Cliffs N. J., 1970

[5] JS EMC DT-R: Mikroprogramy. Wyd. 4, ML-2045901-2. Wyd. Centrum MERA-ELWRO

[6] JS EMC DR-R: Pamięć stała. Wyd. 3, WD-2077001-8. Wyd. Centrum MERA-ELWRO

[7] JS EMC DR-R: Struktura logiczna. Wyd. 3, OL-2045901-2, Wyd. Centrum MERA-ELWRO

[8] Kamburelis T.: Architektura logiczna EMC JS. Rozdział 9: Zgodność programowa i emulacja EMC ODRA 1305. Problemy Informatyki, Warszawa, 1976

[9] Madnick S. E., Donovan J. J.: Operating Systems. Appendix B: Introduction to Microprogramming. Mac Graw-Hill, New York, 1975

[10] MERA-ELWRO: Komputery ODRA w JS EMC. Informator dla użytkowników komputerów Jednolitego Systemu, Wrocław grudzień 1972

[11] Popiel J.: Symulacja maszyn cyfrowych w systemie 3SM. „Organizacja maszyn cyfrowych i mikroprogramowanie", tom II, PWN, Warszawa, 1977

[12] Pulczyn W., Sochacki J.: 3SM System Symulacji Maszyn Mikroprogramowanych. W „Organizacja maszyn cyfrowych i mikroprogramowanie", tom II, PWN, Warszawa, 1977

[13] Rosin R. F.: Contemporary Concepts of Microprogramming and Emulation. ACM Computing Surveys, vol. 1, No. 4, 1969

[14] Wilkes M. V.: The Growth of Interest in Microprogramming. A Literature Survey. ACM Computing Surveys, vol. 1, No. 3, September 1969.

Łamy INFORMATYKI już od roku nie prezentowały problematyki baz danych. Temat ten jest jednak nadal bardzo ważny, szczególnie przy organizowaniu przetwarzania dużych ilości informacji. Co więcej — biorąc pod uwagę duże braki w sprzęcie komputerowym o średniej i dużej mocy, efektywne wykorzystanie istniejących instalacji ma wielkie znaczenie.

W kolejnych numerach INFORMATYKI przedstawiamy dwa artykuły zajmujące się bazami danych. Artykuł Waldemara Kapuścika jest rozwinięciem prac autora z zakresu eksploatacji bazy danych DMS-100 — przedstawionej w INFORMATYCE nr 1/82. Tutaj omówiono projektowanie sieciowych baz danych. Drugi artykuł będzie omawiał (w dwóch częściach) zarządzanie współbieżną aktualizacją bazy danych, a więc prezentował najnowsze zagadnienia w tej dziedzinie. Dla przypomnienia zaś historii rozwoju dziedziny, polecamy opracowanie z INFORMATYKI nr 10 z roku 1983. (Red.)

WALDEMAR KAPUŚCIK

Huta Metali Nieżelaznych „Szopienice"

Projektowanie sieciowych baz danych na przykładzie SZBD-1100

Projektowanie bazy danych polega na odpowiednim zorganizowaniu danych na potrzeby konkretnego systemu informatycznego. Zasadniczo istnieją dwie częściowo rozłączone drogi prowadzące do zdefiniowania bazy danych. Są to mianowicie:

— analiza istniejących danych (w kartotekach niekomputerowych, zbiorach tradycyjnych itp.), ich związku między sobą, wielkości, typu i liczby

— analiza projektowanych programów, ich potrzeb, powiązań i ograniczeń.

Obie drogi mogą doprowadzić do optymalnego celu, jakim jest efektywna w wykorzystaniu baza danych. Wydaje się jednak, że znacznie większy wpływ na optymalny kształt bazy ma analiza przyszłych programów systemu. To pod kątem zastosowanych w nich algorytmów powinno się definiować optymalne struktury danych — oczywiście optymalne dla danego programu czy systemu użytkowego. Optymalność oznaczą tu minimalizację czasu wykonania transakcji korzystających z bazy. Problem zajętości pamięci masowej, aczkolwiek też ważny, winien stać na drugim miejscu, zwłaszcza w bazach danych dla systemów czasu rzeczywistego.

Mówiąc o projektowaniu bazy danych mamy na myśli zdefiniowanie określonych obiektów czy struktur dopuszczalnych w danym systemie zarządzenia Bazą Danych (SZBD). W zależności od struktury baz danych (relacyjne, hierarchiczne, sieciowe), słowo **obiekt** może reprezentować różne fizyczne i logiczne pojęcia. W relacyjnych bazach danych podstawowymi obiektami są relacje i tworzące je atrybuty; w bazach hierarchicznych — fizyczne i logiczne segmenty główne i podporządkowane; w bazach sieciowych — rekordy, obszary i grupy logiczne łączące rekordy powiązane ze sobą w określony sposób. W każdym z tych przypadków projektowanie bazy oznacza co innego, zawiera inne problemy i inne są metody ich rozwiązywania.

Projektowanie relacyjnych baz danych polega na wyborze atrybutów tworzących relację. Aby mówić o właściwym wyborze, należy zbadać wszystkie rzeczywiste związki za-

chodzące między tymi atrybutami (zależność funkcjonalna tranzytywna, wielowartościowa). Uproszczenie lub ominięcie tych związków w relacjach pozwoli na doprowadzenie relacji do trzeciej lub czwartej postaci normalnej, co z punktu widzenia efektywności przetwarzania jest najlepszym rozwiązaniem. Inne problemy, jakie można tu spotkać, to wybór kluczy głównych i kandydujących relacji, a także — definiowanie relacji zewnętrznych, wywodzących się z modelu pojęciowego za pomocą rachunku predykatów czy algebry relacji [1].

Na inne problemy można natknąć się przy projektowaniu hierarchicznych baz danych. Podejście to narzuca grupowanie elementarnych pól w segmenty (odpowiedniki rekordów), wśród których wyróżniamy segmenty główne i podporządkowane. Segmenty te mogą być powiązane ze sobą wskaźnikami hierarchicznymi i wskaźnikami kolejnych segmentów bliźniaczych (jedno- i dwukierunkowymi). Można tu mówić zarówno o fizycznej, jak i o logicznej bazie danych (złożonej z jednej lub kilku baz fizycznych). Co więcej, w podejściu hierarchicznym można definiować nawet związki typu „n:m" poprzez dwukierunkowe związki logiczne zrealizowane za pomocą łączenia fizycznego lub wirtualnego. Wreszcie można wybrać określone metody dostępów do segmentów (np. w systemie IMS są nimi organizacje HSAM, HISAM, HDAM i HIDAM) [1].

Jeszcze inne obiekty definiujemy w sieciowym modelu danych. Podstawowym z nich jest rekord składający się zarówno z danych użytkowych (pole danej), jak i wskaźników łączących ten rekord z innymi rekordami bazy (pole wskaźnika). Związki logiczne wiążące rekordy różnych typów mogą być zrealizowane poprzez grupy logiczne (ang. set), zorganizowane w różny sposób i za pomocą różnych wskaźników („w przód", „wstecz", „do właściciela"). Z kolei rekordy mogą być umieszczane w różnych fizycznie obszarach i można dla nich określać różne metody alokacji.

Projektowanie sieciowej bazy danych można właściwie podzielić na dwie fazy. Pierwszą z nich, mającą większy wpływ na końcowy wynik, jest projektowanie struktury logicznej bazy. Ta faza projektowania daje odpowiedzi na dwa pytania: jak dane powiązać między sobą oraz jak je pogrupować w rekordy. Faza druga to projektowanie struktury fizycznej bazy, a więc powiązanie danych z konkretnymi fizycznymi urządzeniami; określa to wielkość bazy, metody dostępu i sposoby zabezpieczania danych. Poniższe uwagi dotyczyć będą sieciowych baz danych opartych na zasadach określonych przez komitet CODASYL, a zaimplementowanych w systemie DMS-1100 na komputerach serii UNIVAC 1100 [2].

PROJEKTOWANIE STRUKTURY LOGICZNEJ BAZY DANYCH

Projektowanie logicznej struktury bazy danych jest procesem złożonym i wieloetapowym. Można je podzielić na trzy etapy — projektowanie wstępnej struktury logicznej, ocena tej struktury i jej ulepszanie. Projektowanie struktury wstępnej to całość prac projektanta (administratora) bazy od momentu zdefiniowania funkcji systemu, poprzez budowę najprostszych struktur danych realizujących pojedynczą funkcję, aż po scalenie ich w jedną bazę danych. Etap ten może więc być podzielony następująco: zbieranie informacji od użytkownika, definiowanie i grupowanie funkcji systemu, analiza i konsolidacja danych, definiowanie rekordów i grup logicznych.

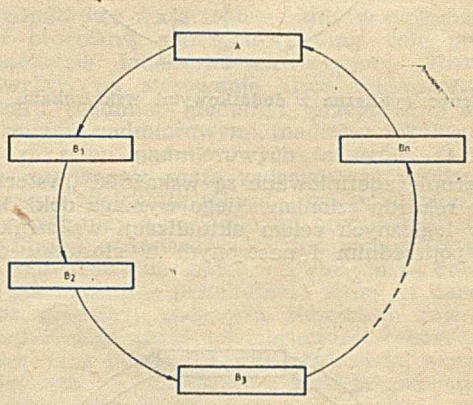
W czasie zbierania informacji od użytkownika, należy pamiętać o precyzowaniu żądań pod adresem systemu oraz określaniu danych liczbowych charakteryzujących zarówno same dane (typ, liczbę, wielkość), jak i funkcje je przetwarzające (częstotliwość).

O łączeniu danych w jeden rekord decydować mogą różne czynniki. Niektóre z nich są oczywiste, jak zależność typu „1:1" czy konieczność bezpośredniego dostępu według różnych pól kluczowych wymuszająca rozdzielenie obu kluczy. Inne kryteria, które powinny być brane pod uwagę, to wielkość pola i częstotliwość jego wykorzystania przez transakcje systemu. Jest intuicyjnie wyczuwalne, że pola duże i rzadko używane winny być umieszczone poza rekordem głównym, a więc powinny stanowić niejako przedłużenie (kontynuację) rekordu głównego. Zmniejszy to czas transmisji całego rekordu i może wpłynąć na zmniejszenie programu (poprzez zmniejszenie buforów na rekordy). Innymi czynnikami, które mogą być brane pod uwagę, są: ochrona danych przed niepowołanym dostępem, ograniczenia sprzętowe, stabilność danych, czy potrzeby przyszłej reorganizacji.

O łączeniu rekordów w grupy logiczne decyduje przede wszystkim zależność typu „1:n" między rekordami. Tymczasowy lub stały charakter związku implikuje zdefiniowanie grupy jako manualnej lub automatycznej. W grupie można definiować różnego rodzaju wskaźniki; na etapie projektu wstępnego wybieramy najprostsze rozwiązanie — wskaźniki „w przód".

Z uwagi na to, że SZBD dostarczają przeważnie wielu różnych możliwości definiowania i grupowania danych w bazie, istnieje potrzeba znalezienia w miarę obiektywnej metody porównywania alternatywnych rozwiązań. Metodą taką może być metoda obliczania liczby logicznych dostępów do bazy danych [3]. Przez dostęp logiczny do rekordu rozumiemy przeczytanie tego rekordu przez SZBD, celem pobrania danych użytkowych lub wskaźników do innych rekordów bazy. O wyborze logicznych, a nie fizycznych, dostępów do bazy zadecydowało to, że na etapie projektowania jest prawie niemożliwe określenie liczby fizycznych dostępów, a do jej wyliczenia potrzebna byłaby znajomość rozmieszczenia wszystkich rekordów w bazie (mapa pamięci).

Metoda obliczania liczby logicznych dostępów polega na rozbiciu każdej transakcji na skończoną liczbę operacji elementarnych, takich jak: odczyt rekordu, zapis rekordu, usunięcie rekordu i jego modyfikacja. Znając koszt pojedynczej operacji w zadanej strukturze danych [3], można obliczyć koszt pracy całej transakcji. Mówiąc **koszt**, mamy na myśli liczbę logicznych dostępów do bazy. W przypadku podobnych kosztów całkowitych, przy wyborze właściwego rozwiązania można kierować się innymi względami, choćby zajętością pamięci masowej.



Rys. 1. Grupa cykliczna ze wskaźnikami „w przód" (A — właściciel grupy, Bi — rekordy członkowskie grupy)

Działanie metody obliczania liczby logicznych dostępów pokażemy na możliwie najprostszych przykładach. Pierwszy z nich to obliczenie kosztu odczytu dowolnego rekordu z grupy cyklicznej (rys. 1). Niech w każdym wystąpieniu tej grupy będzie średnio „n" rekordów. Aby przeczytać dowolny rekord tej grupy należy przede wszystkim wybrać właściwe wystąpienie tej grupy poprzez ustalenie rekordu właściciela (rekord A). Następnie, aby przeczytać pierwszy rekord (B1), należy dodatkowo wykonać jeden dostęp, drugi — dwa dostępy itd. Załóżmy, że prawdopodobieństwo wyboru rekordu z tej grupy jest takie samo dla wszystkich rekordów i wynosi „1/n". Niech X(1,2,3...n) oznacza zmienną losową liczby dostępów do rekordów członkowskich grupy. Wtedy średnia liczba logicznych dostępów potrzebna do odszukania dowolnego rekordu wynosi

$$K = EX = \frac{1}{n}(1+2+\ldots+n) = \frac{n+1}{2}$$

Jak widać, koszt całkowity to jeden dostęp do rekordu typu „właściciel grupy" i (n+1), a dwa dostępy do rekordów typu „członek grupy".

Innym przykładem może być obliczenie kosztu usunięcia rekordu członkowskiego z grupy cyklicznej. Załóżmy, że w każdym wystąpieniu grupy jest średnio „n" rekordów o jednakowym prawdopodobieństwie odczytu. Koszt usunięcia rekordu członkowskiego ma dwie składowe — koszt znalezienia tego rekordu i koszt reorganizacji grupy. Zwłaszcza ta druga składowa silnie zależy od rodzaju wskaźników używanych w grupie.

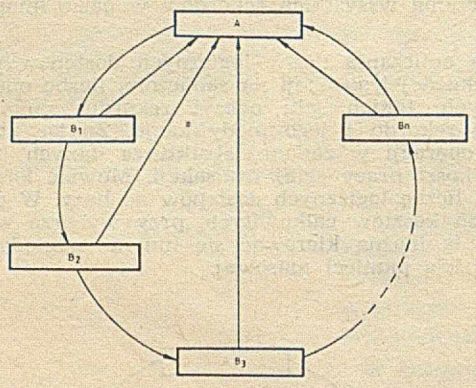
Załóżmy najpierw, że w grupie zdefiniowano jedynie wskaźniki „w przód". Usunięcie j-tego rekordu tej grupy

będzie polegało na aktualizacji wskaźnika „w przód" w rekordzie „j—1". Z braku wskaźników „wstecz" należy więc wykonać:

$$n-j+1+j-1=n$$

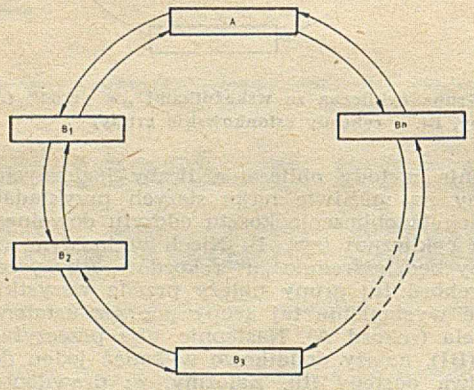
dostępów, a w rozbiciu na typy rekordów: jeden dostęp do rekordu właściciela i n—1 dostępów do rekordów członkowskich. Załóżmy z kolei, że w grupie zdefiniowano również wskaźniki „do właściciela" (rys. 2). Wtedy koszt reorganizacji zmniejszy się, gdyż bezpośrednio z rekordu „j" można przejść do rekordu właściciela. Wartości zmiennej losowej X(0,1...n—1) przedstawiają liczby koniecznych dostępów (nie licząc dostępu do właściciela). Średnia ich liczba wyniesie więc:

$$K = EX = \frac{1}{n}(0+1+\ldots+n-1) = \frac{n-1}{2}$$



Rys. 2. Grupa cykliczna z dodatkowymi wskaźnikami „do właściciela"

Gdy w grupie zdefiniowane są wskaźniki „wstecz" (rys. 3), usunięcie rekordu członkowskiego wymaga dokładnie dwóch dostępów logicznych celem aktualizacji wskaźników w rekordach poprzednim i następnym w stosunku do usuwanego.



Rys. 3. Grupa cykliczna z dodatkowymi wskaźnikami „wstecz"

Podobne obliczenia można wykonać na bardziej skomplikowanych strukturach, aż do pełnej struktury sieciowej włącznie. Można więc w ten sposób obliczać koszt wstawienia rekordu do jednej czy kilku grup cyklicznych, koszt usunięcia rekordu członkowskiego z wielu grup, czy też koszt usunięcia rekordu właściciela w zależności od tego, jak powiązane są jego rekordy członkowskie z innymi rekordami bazy. Podobne obliczenia, aczkolwiek prostsze, wykonuje się dla innych typów grup logicznych, np. dla grup z tablicami wskaźników (ang. pointer array set).

Przedstawiona metoda może służyć do oceny wstępnego projektu struktury logicznej bazy danych. Optymalizację jednego programu (funkcji) uzyskuje się poprawiając strukturę danych tak, aby zmniejszyć liczbę logicznych dostępów tej transakcji. Łatwo jednak zauważyć, że w złożonej strukturze sieciowej poprawienie struktury dla jednego programu może pogorszyć, nawet znacznie, warunki pracy innych programów. W związku z tym proponuje się minimalizację kosztu pracy całego systemu, który można zdefiniować następująco:

$$K = \sum_{i=1}^{n} pi\ Ti$$

gdzie: n — liczba transakcji w projektowanym systemie; pi — częstotliwość i-tej transakcji systemu (w %); Ti — średni koszt pracy i-tej transakcji systemu (liczba logicznych dostępów).

Rozwiązaniem optymalnym byłaby taka struktura danych, dla której powyższa forma liniowa przyjmowałaby minimum. Analitycznie nie można wyznaczyć tego minimum, ale przedstawimy metodę pozwalającą na zbliżenie się do niego. Metoda polega na poprawie struktury dla transakcji najczęstszych i najbardziej pracochłonnych. Zauważmy, że udział transakcji w obciążeniu całego systemu odzwierciedla iloczyn „pixTi", który jest nazywany **współczynnikiem obciążenia transakcji w systemie.**

W metodzie tej wszystkie transakcje systemu grupujemy wyróżniając wśród nich:

1) programy w czasie rzeczywistym o wysokiej częstotliwości realizacji
2) programy w czasie rzeczywistym o małej częstotliwości realizacji, lecz dużym wskaźniku obciążenia.
3) programy wsadowe o dużej częstotliwości realizacji.
4) inne programy, dla których nie poprawiamy struktury danych.

W ostatniej grupie znajdą się te programy, których udział w systemie jest niski lub których nie można przyspieszyć poprzez poprawę struktury danych. Do grupy tej można zaliczać programy według następującego kryterium:

— program przetwarza rekordy tylko jednego typu
— wskaźnik obciążenia programu w czasie rzeczywistym <50
— częstotliwość realizacji programu wsadowego $<5\%$.

Dla grup 1, 2 i 3 można poprawić strukturę danych. Grupy te mają coraz to niższy priorytet, co oznacza, że w razie konfliktu rozwiązań dla dwóch różnych transakcji wybieramy rozwiązanie poprawiające pracę transakcji o wyższym priorytecie. W razie konfliktu transakcji o tym samym priorytecie, o wyborze rozwiązania decydują inne czynniki, jak wyższy wskaźnik obciążenia czy większa oszczędność pamięci masowej.

Pokażemy teraz kilka typowych metod poprawy struktury danych dla pojedynczej transakcji.

● **Dostęp w górę** (ang. upward access) odbywa się wtedy, gdy poprzez dostęp bezpośredni i sekwencyjne przeszukiwanie obszaru lub grupy pobraliśmy pewien rekord bazy, a następnie wymagane jest pobranie informacji z rekordu stojącego wyżej w strukturze. Z reguły jest to rekord właściciela grupy (innej niż użyta do znalezienia rekordu), w której rekord bieżący jest członkiem. Możliwe są dwa rozwiązania poprawiające taką strukturę. Pierwsze z nich — to celowa redundancja danych, a więc dołożenie do rekordu członkowskiego tych pól, po które najczęściej odbywał się dostęp w górę. Drugim rozwiązaniem jest dołożenie do rekordu członkowskiego kodu właściciela (o ile jest do niego dostęp bezpośredni), adresu lub dodatkowego wskaźnika „do właściciela" w tej grupie, przez co zminimalizuje się liczbę logicznych dostępów. O wyborze właściwego rozwiązania decyduje wielkość klucza właściciela w porównaniu z wielkością klucza bazy danych (wskaźnika).

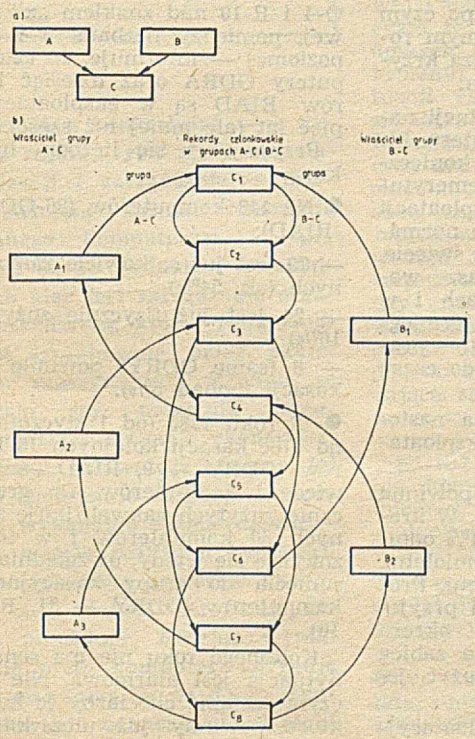
● **Dostęp w dół** (ang. downward access) jest bardzo często realizowany w sieciowych bazach danych, gdyż właśnie w tym celu definiowane są grupy logiczne. Tym niemniej można się zastanawiać, czy zawsze grupa jest potrzebna oraz jak usprawnić i przyspieszyć tę formę przetwarzania. Jednym z rozwiązań jest konsolidacja danych na wyższym poziomie w rekordzie stałej lub zmiennej długości, a więc faktycznie likwidacja grupy. Rozwiązanie to można stosować, gdy liczebność grupy jest niezbyt duża i w miarę stabilna. Inne rozwiązanie może być zastosowane wtedy, gdy przetwarzanie w dół odbywa się w celu skumulowania pewnych wielkości z niższego poziomu hierarchii. Wtedy zdefiniowanie w rekordzie-właścicielu pola, zawierającego sumy wszystkich wartości z rekordów członkowskich, znacznie przyspieszy przetwarzanie. Czynnikiem usprawniającym przetwarzanie w dół jest również właściwe zdefiniowanie porządku w grupie (tzn. logicznego punktu wstawienia) i rodzaju wskaźników w grupie.

● **Usuwanie rekordów z bazy danych** jest najbardziej kosztowną operacją na bazie danych, gdyż powoduje konie-

czność aktualizacji wszystkich łańcuchów, w których miał miejsce usuwany rekord. Koszt usuwania rekordu zmniejszymy poprzez dodanie wskaźnika „wstecz" do wszystkich łańcuchów, w których rekord uczestniczy lub przynajmniej do łańcuchów najliczniejszych.

● **Dostęp do informacji w prostej strukturze sieciowej** (rys. 4) odbywa się wtedy, gdy rekord jest jednocześnie członkiem dwóch różnych grup. Typowe przetwarzanie takiej struktury polega na dostępie w dół, wzdłuż jednej grupy i dla każdego rekordu członkowskiego pobieranie informacji z rekordu właściciela grupy drugiej. Przetwarzanie tego typu można usprawnić przez dodanie wskaźników „do właściciela" w jednej lub obu wymienionych grupach.

Poprawa struktury logicznej bazy danych powinna zakończyć się obliczeniem kosztu pracy całego systemu, a więc kosztu pracy wszystkich transakcji, nawet tych, dla których nie poprawialiśmy struktury danych. W poszczególnych przypadkach koszt ten może nawet poważnie wzrosnąć na skutek zmian struktury dla innych transakcji, co jednak jest wkalkulowane w koszty metody.



Rys. 4. Prosta struktura sieciowa — rekord C zdefiniowany jako członek dwóch grup cyklicznych; a) zapis symboliczny, b) fizyczna realizacja sieci

PROJEKTOWANIE STRUKTURY FIZYCZNEJ

Projektowanie struktury fizycznej bazy danych można uznać za powiązanie danych z fizycznymi urządzeniami — zbiorami w pamięci masowej. Rzecz więc w obliczeniu i wyborze parametrów obszarów danych, takich jak wielkość strony, liczba stron i struktura strony z danymi. Na tym etapie należy również zadecydować, jakie metody bezpośredniego dostępu z dopuszczalnych przez SZBD przydzielić poszczególnym typom rekordów oraz jaki rodzaj stron nadmiarowych zdefiniować w każdym obszarze.

Prawidłowy wybór parametrów struktury fizycznej bazy zapewnia jej efektywną eksploatację. Koszt pracy całego systemu zależy w głównej mierze od poprawnego projektu struktury logicznej, jednakże i we właściwym zdefiniowaniu fizycznej bazy danych tkwią potencjalne możliwości skrócenia czasu pracy programów. Należy dodać, że w odróżnieniu od poprzedniego etapu, projektowanie struktury fizycznej zależy w dużym stopniu zarówno od samego SZBD, jak i parametrów sprzętu, jego ograniczeń i możliwości.

Podstawowym problemem w fazie projektowania fizycznej bazy danych jest łączenie rekordów w obszary. SZBD pozwalają tutaj na dużą swobodę. Rekordy tego samego typu mogą być przechowywane w różnych obszarach; w jednym obszarze mogą być przechowywane rekordy różnych typów, stałej i zmiennej długości, nawet o różnych metodach alokacji. O zapamiętaniu różnych typów rekordów w jednym obszarze winny decydować następujące czynniki: typ przetwarzania danych, ochrona danych przed niepowołanym dostępem, zabezpieczenie integralności danych, wielkość rekordów w porównaniu z wielkością stron, metody rozwiązywania konfliktów oraz częstość i przebieg reorganizacji. Rekordy częściej używane i reorganizowane winny być umieszczane, w miarę możliwości, w odrębnych obszarach i to przydzielonych do najszybszych urządzeń zewnętrznych (pamięci).

Wielkość strony obszaru bazy danych zależy od wielkości buforów roboczych SZBD, typu pracy systemu (wieloprogramowo czy nie) oraz sposobu przetwarzania stron (sekwencyjnie czy wyrywkowo). Praca wieloprogramowa i przetwarzanie w sposób bezpośredni wymusza na projektancie stosowanie jak najmniejszych stron, przez co skraca się czas transmisji danych i zmniejsza prawdopodobieństwo blokowania tych samych stron przez różne programy.

Wielkość obszaru zależy w oczywisty sposób od planowanej liczby rekordów, które w tym obszarze będą pamiętane. Znając wielkość rekordu lub średnią wielkość rekordu w obszarze, wielkość stron i liczbę rekordów, łatwo obliczyć wymaganą liczbę stron. Przy obliczaniu tej liczby projektant powinien wziąć poprawkę w górę — umożliwiającą ekspansję systemu oraz ułatwiającą pracę procedurom alokacji. Sprawność wielu procedur alokacji zależy bowiem od stopnia załadowania obszaru; przy dużym stopniu załadowania coraz to więcej rekordów znajdzie się na stronach obcych lub nadmiarowych.

Strony nadmiarowe mają zastosowanie w metodzie alokacji poprzez procedurę obliczeniową (ang. calc procedure) oraz w metodzie indeksowo-sekwencyjnej. Poszczególne SZBD pozwalają na stosowanie stron nadmiarowych na końcu obszaru (nadmiar globalny), przeplatanie stron na dane ze stronami nadmiarowymi (nadmiar lokalny) lub na rezygnację ze stron nadmiarowych w ogóle. O wyborze właściwego rozwiązania może zadecydować wielkość stron w porównaniu z wielkością rekordu, typ przetwarzania oraz możliwość współpracy z posiadanymi procedurami randomizacyjnymi. Strony nadmiarowe lokalne nadają się szczególnie dla małych stron (lub takich, na których są duże rekordy) i stanowią naturalne przedłużenie stron na dane. Z kolei nadmiar globalny, zwłaszcza stosowany równocześnie z zasadą najbliższego miejsca (umieszczanie rekordów członkowskich możliwie blisko rekordu właściciela, a więc również na stronach nadmiarowych), sprzyja sekwencyjnemu przetwarzaniu dużych stron.

Metody dostępu do rekordów, a zwłaszcza metody dostępu bezpośredniego, zależą od możliwości SZBD. System DMS-1100 dopuszcza stosowanie trzech rodzajów dostępu bezpośredniego: za pomocą adresu (DIRECT), poprzez procedurę randomizacyjną (CALC) i indeksowo-sekwencyjny. Wybór metody zależy od liczby rekordów, ich stabilności, budowy i znajomości kluczy lub co najmniej ich zakresów wartości oraz typu przetwarzania. Najszybszą i najlepiej wykorzystującą pamięć metodą jest dostęp DIRECT, lecz wymaga ona znajomości przez administratora adresów wszystkich rekordów tego typu w bazie. Nieco wolniejszy, gorzej wykorzystujący pamięć jest typ dostępu CALC, nadający się raczej do przetwarzania wyrywkowego, lecz nie wymagający dokładnej znajomości rozkładu wartości kluczy. Z kolei metoda indeksowo-sekwencyjna nadaje się raczej do dużych jednorodnych kartotek rekordów w miarę stabilnych, a więc niezbyt często wstawianych, usuwanych czy zmieniających klucz.

Niektóre SZBD pozwalają na wybór kilku standardowych procedur randomizacyjnych (DMS-1100 ma procedury DMSCALC i RANDENTIAL) lub na łatwe włączenie do systemu procedury własnej. Podstawowym kryterium przydatności danej metody powinny być: czas dostępu, liczba kolizji i średnia liczba rekordów nadmiarowych. Można badać, na ile dana funkcja mieszająca odbiega od ideału (w sensie minimalizacji czasu dostępu) poprzez porównanie przebiegu tej funkcji z przebiegiem granicznym idealnej funkcji mieszającej [4]. Teoretycznie, dla różnych typów rekordów, a właściwie dla różnych kluczy można opracować oryginalne procedury obliczeniowe, dające bardzo dobre rozkłady wartości klucza w zadanej przestrzeni adresowej. Procedury te łatwo włączyć do SZBD, ale należy pamiętać o tym, że całkowita wielkość SZBD istotnie wzrośnie.

Przedstawioną metodę projektowania sieciowych baz danych zastosowano z powodzeniem w hucie „Szopienice". Skrócenie czasu pracy transakcji już na etapie projektowania zwiększyło przepustowość i możliwości posiadanego systemu komputerowego. W rezultacie w Hucie eksploatuje się obecnie osiem systemów użytkowych, a nie — jak to pierwotnie zakładano — tylko jeden. Dodajmy jeszcze, że wszystkie systemy użytkowe są eksploatowane w trybie czasu rzeczywistego, a dane do nich wprowadzane są z terminali umieszczonych bezpośrednio na stanowiskach produkcyjnych.

LITERATURA

[1] Date C.: Wprowadzenie do baz danych. WNT, 1981

[2] Kapuścik W.: Projektowanie baz danych typu CODASYL. Skrypt Politechniki Śląskiej, 1983

[3] Kapuścik W.: Liczba logicznych dostępów w bazach danych typu CODASYL. Podstawy sterowania, t. 10, z. 2, 1980

[4] Konopacki A., Piwowarski M., Wełnicki L.: Metodyka analizy przydatności i doboru funkcji mieszających. ZSAK PAN, Gliwice, 1979.

Z kraju

Komputery w „N-tym" resorcie

Kryzys w gospodarce, kryzys w informatyce. Zbieżność kryzysów nie dziwi, bo przecież informatyka (w sensie zastosowań i obsługującego je sprzętu) jest pochodną, funkcją stanu gospodarki narodowej.

Główną ofiarą kryzysu w informatyce jest jej wyposażenie techniczne — komputery. Dekapitalizują się! Nie chcę operować średnim wiekiem obecnie eksploatowanych maszyn lub innymi zagregowanymi danymi statystycznymi dotyczącymi kraju, czy też innego kryterium odniesienia. Chciałbym pokazać problem dekapitalizacji komputerów na przykładzie jednego z ważnych gospodarczo i — jak na krajowe warunki — stosunkowo dobrze skomputeryzowanego resortu, a w nim komputerów ODRA i RIAD. Komputery te, w liczbie 148, stanowią ponad 80% wszystkich komputerów eksploatowanych w jednostkach tego resortu. Istoty sytuacji nie zmienia fakt, że wśród tych 148 komputerów znajdują się pojedyncze egzemplarze R-10 i R-12, które można zaliczyć do kategorii minikomputerów. Ostrość problemu ilustruje załączony rysunek, który wymaga jednak skomentowania.

Na rysunku pokazano krzywą sprawności eksploatacyjnej komputerów ODRA i RIAD (1 stycznia 1984). Przy jej tworzeniu przyjęto, że do pełnej sprawności eksploatacyjnej komputery dochodzą po około dwóch latach eksploatacji, a także — że maksymalna sprawność eksploatacyjna utrzymuje się do 5—6 roku eksploatacji, po czym zmniejsza się powoli, a po ósmym roku — gwałtownie. Ponadto (poza krzywą) przyjęto poniższe założenia:

- Po dziesięciu latach eksploatacji następuje fizyczne zużycie komputera. Fizyczne zużycie to sygnał o konieczności przejścia maszyny na „emeryturę". Wiemy, że dziesięć lat eksploatacji komputera znacznie przekracza normatywy powszechnie przyjęte na świecie. Trzeba jednak uwzględniać nasze warunki. Na zwolnionych obrotach i w sprzyjających warunkach niektóre egzemplarze mogą — inne muszą — pracować jeszcze na „pół etatu", do czasu kasacji.
- Kasacja komputera powinna następować po dwunastu latach eksploatacji.
- Amortyzacja komputera powinna przebiegać w ciągu siedmiu lat. W kraju obowiązywało poprzednio 14% odpisu rocznie, a więc okres siedmioletni, zgodny ze specyfiką tego sprzętu. Procesu dekapitalizacji nie zmieni przyjęcie obecnie dziesięcioletniego okresu amortyzacji — jest to jedynie zabieg buchalteryjny, który może służyć jedynie poprawie samopoczucia.

Na krzywej sprawności eksploatacyjnej znajdują się znaki „x". Są one umieszczane w połowie odcinków, nad liczbami opisującymi na osi poziomej czas eksploatacji komputerów w latach. Jest tak dlatego, ponieważ w statystyce wiek komputera jest ujęty w postaci „ponad n lat", np. ponad siedem lat, co równocześnie oznacza, że mniej niż osiem lat. Oznaczenia nad znakami „x" na krzywej informują, ile komputerów ODRA i RIAD ma określoną „wysługę lat". I tak np. opis: O-4 i R-10 nad znakiem „x" na krzywej, pomiędzy liczbami 5 i 6 na osi poziomej — informuje, że cztery komputery ODRA oraz dziesięć komputerów RIAD są w eksploatacji ponad pięć lat (ale mniej niż sześć).

Przyjrzyjmy się liczbom na rysunku:

- Na 148 komputerów (86-ODRA i 62-RIAD):

— 79 jest już całkowicie zamortyzowanych (ok. 54%)

— 24 jest już fizycznie zużytych (ok. 16%)

— 8 (same ODRY) powinno już ulec kasacji (ponad 5%).

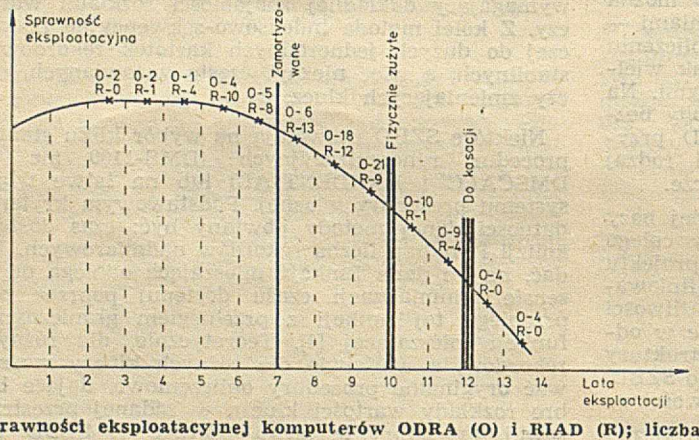
- W roku 1985 (od 1 stycznia) powinno ulec kasacji kolejnych 13 komputerów (ODRY — 9, RIAD — 4), razem więc 21 komputerów. Do grupy fizycznie zużytych zakwalifikuje się kolejnych 30 komputerów i w tej grupie znajdzie się wtedy, uwzględniając przesunięcia do grupy kasacyjnej — 41 komputerów (ODRY — 31, RIADY — 10).

Kolejnego roku nie ma sensu liczyć. Sytuacja jest alarmowa. Nie wiadomo czym zastąpić chociażby te komputery, które powinny już ulec kasacji lub zbliżają się do tego stanu. Sytuację sprzętową komplikuje dodatkowo fakt, że wśród zakładowych ośrodków obliczeniowych przeważają ośrodki wyposażone w jeden komputer.

Na 21 komputerów, które powinny ulec kasacji w latach 1984—1985 są 4 komputery RIAD i 17 komputerów ODRA. Załóżmy, że 4 nowe RIADY zastąpią te stare i wysłużone. A czym zastąpić wysłużone ODRY? Jeśliby się nawet na ten cel przeznaczyło kilka nowych komputerów RIAD (sygnalizowana produkcja ELWRO na te lata jest bardzo mała), to co zrobić z wykorzystanym oprogramowaniem, jeśli emulator ODRA-RIAD jest zapowiedziany dopiero na rok 1987 (praktyczna eksploatacja najwcześniej w 1988 roku). A reszta? Może SM? — a jeśli tak, to co z ich oprogramowaniem aby mogły one przejąć funkcje skasowanych maszyn ODRA...

Taka jest sytuacja, takie wątpliwości i niepewności, dotyczące „N"-tego resortu. A przecież dekapitalizacja sprzętu dotyczy również innych resortów, całej polskiej informatyki.

Słychać, że tu i ówdzie zamierza się, a nawet rozpoczyna opracowywa-



Krzywa sprawności eksploatacyjnej komputerów ODRA (O) i RIAD (R); liczba egzemplarzy wg stanu na 1 stycznia 1984

nie programów (nie prognoz — programów!) rozwoju informatyki do 1995 czy nawet 2000 roku. Przedstawiony przykład wskazuje raczej na potrzebę innego działania. Z pracą nad wieloletnim programem rozwoju informatyki w resorcie proponuję kilka lat poczekać, a wysiłek i inwencję odpowiednich instytucji i osób skierować niezwłocznie na zaprogramowanie konkretnych przedsięwzięć i rozwiązań — w okresie najbliższych 3—4 lat — mających na celu w pierwszym rzędzie zahamowanie dekapitalizacji podstawowego sprzętu informatycznego.

W pierwszym kroku należy opracować realny program przełamania opisanego kryzysu. Kolejny krok może nastąpić dopiero wtedy, gdy będą widoczne skutki początkowego działania. Trzeba to zrobić uwzględniając obecne realia.

Zakończę truizmem. Wprowadzenie informatyki i komputera do przedsiębiorstwa to sprawa trudna, kłopotliwa i kosztowna, niekiedy swoista „droga przez mękę". Ale gdy instrumenty te już działają i uzyskały powszechną akceptację, to bez nich przedsiębiorstwo nie potrafi już poprawnie funkcjonować.

H.Z.

Gra finansowa na MERZE 400

W ekonomii, a konkretniej — w dyscyplinach zajmujących się problemami organizacji i zarządzania, najbardziej rozwiniętą formą nauczania wspomaganego komputerem są gry kierownicze [2, 3]. Zadaniem uczestników takich gier jest zarządzanie organizacją gospodarczą, której funkcjonowanie jest symulowane przez pakiet programów maszynowych. Zgodnie ze scenariuszem gry, uczestnik samodzielnie analizuje sytuację ekonomiczną organizacji, przewiduje kierunki jej rozwoju i podejmuje decyzje gospodarcze. Po wykonaniu eksperymentu — obserwuje wpływ realizacji podjętych decyzji na dynamikę procesów ekonomicznych, a w efekcie weryfikuje zastosowane procedury. Kolejne wyniki generowane przez komputer korygują działania uczestnika — aż do momentu sformułowania zadowalających reguł postępowania decyzyjnego, reguł prowadzących do spodziewanego wyniku.

W Zespole Symulacji Komputerowej Instytutu Organizacji i Zarządzania Politechniki Wrocławskiej prowadzone są prace badawcze nad metodologią projektowania i eksploatacji komputerowych gier finansowych. Gry finansowe stanowią pewną klasę funkcjonalną gier kierowniczych i związane są z procesami i zjawiskami finansowymi w różnego typu organizacjach gospodarczych.

W ostatnim czasie prace naszego zespołu dotyczyły gry finansowej, której eksploatacja byłaby możliwa przy użyciu minikomputera produkcji krajowej. Rezultatem tych prac było opracowanie i wdrożenie do procesu dydaktycznego wielozespołowej gry finansowej KGF-1 obsługiwanej przez system programów GAME. W swej wersji podstawowej system GAME jest przetwarzany przez minikomputer MERA-400.

Zgodnie z przyjętym programem dydaktycznym gry KGF-1, podstawowym celem uczestników jest zapoznanie się z funkcjonowaniem systemu ekonomiczno-finansowego przedsiębiorstwa działającego w warunkach reformy gospodarczej. Zadaniem uczestnika jest sekwencyjne podejmowanie decyzji dotyczących przyszłej działalności produkcyjnej, inwestycyjnej, finansowej i handlowej przedsiębiorstwa. Decyzje te tworzą zbiór danych początkowych. Rezultatem wykonania eksperymentu jest zestaw informacji o sytuacji ekonomiczno-finansowej przedsiębiorstwa w okresie symulowanym. Informacje te służą uczestnikowi do oceny bieżącej i przewidywanej sytuacji ekonomiczno-finansowej przedsiębiorstwa oraz podjęcia decyzji obowiązujących w następnej symulowanej sekwencji. Uczestnik zmuszony jest do szczegółowej analizy zależności pomiędzy własnymi decyzjami a osiąganym wynikiem; w sposób aktywny — poprzez działania — przyswaja sobie podstawowe reguły i mechanizmy składające się na system ekonomiczno-finansowy przedsiębiorstwa.

KGF-1 jest grą realizowaną w trybie komputerowej symulacji naprzemiennej, przy czym pełny okres symulacji wynosi:

- w wersji długookresowej — piętnaście lat podzielonych na pięć trzyletnich sekwencji decyzyjnych
- w wersji krótkookresowej — dwa lata podzielone na osiem kwartalnych sekwencji decyzyjnych.

Wszelkie czynności związane z organizacją i sterowaniem przebiegu gry są wykonywane przez odpowiednie programy systemu GAME. System ten wykonuje eksperymenty symulacyjne oraz organizuje grę i steruje jej przebiegiem.

Programy obsługujące gry finansowe powinny być modyfikowane, tj. pozwalające na sprawną adaptację programów w przypadku zmiany struktury systemu będącego obiektem symulacji bądź zmiany celu i organizacji przebiegu gry, a zarazem niezawodne, przy dowolnych zestawach danych przygotowanych przez uczestników.

W naszym przypadku niezbędną modyfikowalność uzyskaliśmy przez użycie metody programowania zstępującego [1], czego efektem jest zestaw programów o strukturze modułowej w układzie hierarchicznym. Wyodrębnione na podstawie kryteriów funkcjonalnych moduły cechują silne wewnętrzne powiązania informacyjne, przy stosunkowo słabych powiązaniach z innymi modułami wchodzącymi w skład systemu. Modularna konstrukcja systemu GAME pozwoliła na zastosowania przetwarzania wielonakładowego, co w konsekwencji ograniczyło zapotrzebowanie systemu na pamięć operacyjną do 20 KB. System GAME może być modyfikowany i rozwijany. Charakteryzuje się on zdolnością samodzielnego wykrywania zakłóceń, które sygnalizowane są obsłudze gry przez odpowiednie komunikaty.

Prace rozwojowe nad systemem GAME były prowadzone przez zespoły programistów nie uczestniczące we wstępnej fazie prac projektowych, dlatego też programy zostały napisane przy wykorzystaniu elementów zasad programowania strukturalnego [4].

Komunikacja uczestnika gry z systemem GAME odbywa się za pośrednictwem systemu dialogowego, wchodzącego w skład modułu wprowadzania i kontroli danych wejściowych. Dokonuje on analizy poprawności formalnej wartości zmiennych decyzyjnych wprowadzonych przez uczestnika gry. Jeżeli uczestnik popełni błędy formalne, system sygnalizuje ich charakter i żąda odpowiednich poprawek.

Najbardziej złożoną strukturą informatyczną w systemie GAME jest symulator DEF-4, którego zadaniem jest symulacja wpływu decyzji uczestnika na kształtowanie się wyników ekonomiczno-finansowych w grze. Symulator ten wykonano zgodnie z ideą programowania zstępującego. Tworzy go piętnaście modułów odpowiedzialnych za symulację funkcjonowania poszczególnych mechanizmów systemu ekonomiczno-finansowego przedsiębiorstwa (tworzenie i podział akumulacji, finansowanie środków obrotowych, reprodukcja i finansowanie środków trwałych, itd.). Ponieważ dynamika procesów finansowych związana jest z kształtowaniem się procesów rzeczowych w przedsiębiorstwie, symulator DEF-4 odwzorowuje także wybrane sfery działalności produkcyjnej i handlowej organizacji gospodarczej.

Ze względu na sposób realizacji eksperymentów symulacyjnych, symulator DEF-4 zalicza się do klasy tzw. maszyn sekwencyjnych [5].

Kontrolę poprawności przetwarzania w symulatorze DEF-4 realizuje, podobnie jak w rzeczywistym przedsiębiorstwie, moduł obliczeń rachunkowościowych, składający się z dwóch równoległych submodułów ewidencji operacji finansowych, odpowiednio — na aktywach i pasywach przedsiębiorstwa. W trakcie eksperymentu specjalny podprogram sporządza — dla każdej iteracji obliczeń — bilanse opera-

cyjne symulowanych wielkości i porównuje otrzymane sumy bilansowe. W przypadku wykrycia niezgodności tych sum, generowana jest tabela z wartościami elementów składowych, co znacznie ułatwia lokalizację zakłócenia.

Symulator DEF-4 jest także wyposażony w podprogram eliminacji błędów zaokrągleń. Błędy tego rodzaju mają szczególnie istotny wpływ na poprawność obliczeń w przypadku eksperymentów realizowanych metodą stałego kroku i obsługiwanych przez programy fortranowskie.

Wprowadzenie rejestracji wszystkich wyników pośrednich eksperymentów symulacyjnych na zbiorze dyskowym spowodowało — oprócz zmniejszenia zapotrzebowania na pamięć operacyjną — znaczne rozszerzenie możliwości organizacyjnych gry:

— umożliwienie równoległego uczestnictwa w grze dziewięciu zespołom

— wprowadzenie elastyczności rytmu gry, co oznacza, że w danym momencie czasu rzeczywistego poszczególne zespoły mogą się znajdować w różnych momentach czasu symulowanego

— umożliwienie restartu gry.

Wyniki eksperymentów symulacyjnych zestawia w postaci tabulogramów wyjściowych moduł edycyjny. Podprogramy tego modułu umożliwiają redagowanie wyników eksperymentu w kilkunastu rodzajach zestawień, informujących o sytuacji ekonomiczno-finansowej przedsiębiorstwa. Struktura informacji wyjściowych jest ustalana przez uczestnika gry.

Gra KGF-1 została wdrożona do procesu dydaktycznego na Wydziale Informatyki i Zarządzania Politechniki Wrocławskiej w ramach przedmiotu „Gospodarka finansowa przedsiębiorstw". Przeprowadzona wśród studentów — uczestników gry — ankieta wykazała, iż oceniają oni pozytywnie tę formę dydaktyki. Udział w grze pozwolił na pierwsze praktyczne zastosowanie wiedzy o finansach przedsiębiorstw, nabytej w trakcie wykładów i seminariów.

MIECZYSŁAW BUDZIŃSKI
EDWARD RADOSIŃSKI
LEOPOLD SZCZUROWSKI

Politechnika Wrocławska

LITERATURA

[1] Alagić S., Arbib M. A.: Projektowanie programów poprawnie i dobrze zbudowanych, WNT, Warszawa, 1982
[2] Maclean R.: Simulation and Games, with Particular Reference to the Teaching of Economics. Programed Learning and Educational Technology, vol. 13, No. 3, July 1976, pp. 11—18
[3] Miller R. M.: Oligopoly: a multipurpose computer simulation game. Simulation and Games, vol. 12, No. 4, December 1981, pp. 393—416
[4] Myers G. J.: Projektowanie niezawodnego oprogramowania. WNT, Warszawa, 1980
[5] Ziegler B. P.: Theory of modelling and simulation. J. Wiley and Sons, New York, 1976.

Obsługa techniczna sprzętu w zestawieniu ankietowym

Dotychczas w naszym kraju nie podejmowano badań problemów organizacji i funkcjonowania obsługi technicznej sprzętu komputerowego, mimo że ma ona znaczący wpływ na działalność ośrodków obliczeniowych. Szczególnie dzisiaj, gdy coraz szybciej postępuje proces starzenia się eksploatowanego sprzętu, a liczba nowych instalacji komputerowych jest bardzo mała. W celu zbadania sytuacji, Instytut Przetwarzania Danych i Rachunkowości SGPiS przeprowadził w 1982 roku wielostronną analizę. Jednym z narzędzi badawczych była ankieta, którą objęto pracowników obsługi technicznej 110 ośrodków obliczeniowych w kraju — kierowników wydziału i działu technicznego, specjalistów elektroników, inżynierów, techników, konserwatorów.

W 1983 roku w ośrodkach obliczeniowych zainstalowanych było 2648 komputerów, w tym 840 dużych i średnich oraz 1808 minikomputerów [1]). Wprawdzie ogólna liczba komputerów wzrosła w 1983 roku o 95 sztuk, jednak przeciętny wiek komputera nie zmienił się w stosunku do roku poprzedniego i wynosi siedem lat. Dominującą grupę (ponad 54%) stanowią komputery w wieku od sześciu do dziesięciu lat. Ostatnio znacznie powiększyła się grupa komputerów najstarszych — w wieku od dziesięciu lat wzwyż. Najbardziej niepokoi sytuacja komputerów dużych i średnich: 86,5% ma więcej niż pięć lat, 53% liczy powyżej ośmiu, a ponad 25% osiągnęło już wiek powyżej jedenastu lat. Udział komputerów dużych i średnich mieszczących się w przedziale wieku od jednego roku do trzech lat zmniejszył się z 21,6% w roku 1979 do 3,6% w 1983. Średni wiek minikomputerów wynosi sześć lat i będzie rósł. Minikomputery liczące dziewięć i więcej lat stanowią już ponad 30%.

Proces starzenia się parku komputerowego powoduje sukcesywne zmniejszanie się jego niezawodności, co w konsekwencji zwiększa czas przestoju z przyczyn technicznych. Sytuację dodatkowo pogarsza fakt dużej zawodności produkowanego obecnie sprzętu, a także niska sprawność i efektywność serwisu naprawczego.

Do naprawy, konserwacji i profilaktyki urządzeń informatycznych powołane zostały specjalne służby techniczne — bądź w ramach ośrodków obliczeniowych, bądź w postaci specjalistycznego serwisu fabrycznego. Według danych GUS-u na dzień 31 grudnia 1983, zatrudnionych było 3291 konserwatorów komputerów (w tym 1613 z wyższym wykształceniem) oraz 1500 konserwatorów innych urządzeń informatycznych.

Stan zatrudnienia w obsłudze technicznej budzi wątpliwości. Nadmierne są — w stosunku do innych rodzajów urządzeń technicznych o podobnej technologii — przestoje z przyczyn technicznych, przy dużej liczbie konserwatorów o wysokich najczęściej kwalifikacjach. Obsługa techniczna sprzętu informatycznego jest — jak wszystko wskazuje — prowadzona nieracjonalnie.

Trudna do udowodnienia byłaby teza o rzekomo większej awaryjności sprzętu informatyki w stosunku do innych podobnych urządzeń — zbudowanych na podzespołach elektronicznych, przy użyciu zbliżonych technologii. Wydłużony czas przestojów ma inne przyczyny. Określenie ich, a także ocena racjonalności tak dużego zatrudnienia w obsłudze technicznej, stała się możliwa dzięki przeprowadzonej ankiecie.

Wykształcenie pracowników obsługi technicznej w badanej próbie było następujące:

- wyższe techniczne — 20,5%
- wyższe ekonomiczne — 13,6%
- wyższe matematyczne — 32,7%
- wyższe inne specjalności — 15,5%
- średnie techniczne — 10,9%
- średnie ekonomiczne — 6,8%

Udział personelu z wykształceniem technicznym jest zdumiewająco niski (31,4%), a powinien stanowić grupę dominującą. W sytuacji gdy prawie 70% personelu technicznego stanowią pracownicy o wyuczonym innym profilu zawodowym, istotnym elementem nabywania właściwych kwalifikacji zawodowych są kursy specjalistyczne — krajowe i zagraniczne. Jak wykazało badanie, 92,7% respondentów dodatkowo kształciło się na różnych kursach (zakładowe, specjalistyczne, organizowane przez MERA-ELWRO).

Ankieta potwierdziła, że wszystkie kategorie pracowników obsługi technicznej w równej mierze podlegają szkoleniu na kursach. Dominującą formą jest kształcenie na kursach prowadzonych przez MERA-ELWRO — 79% badanych pracowników obsługi technicznej ukończyło jeden lub więcej takich kursów.

[1]) Informatyka i Ośrodki Informatyki w 1983 roku. GUS, Ośrodek Badawczo-Rozwojowy SPIS, Warszawa, czerwiec 1984

dokończenie na str. 21

Zeszłoroczny cykl publikacji o APPLE IIe wywołał sporo kontrowersji. Wiele osób zarzucało nam marnowanie miejsca mikroKLANU na opisywanie komputera będącego praktycznie poza granicą możliwości finansowych większości czytelników. Z drugiej strony — do autora publikacji (podaliśmy numer telefonu) dzwoniło wiele osób chcących uzyskać bardziej szczegółowe informacje. Często rozważano możliwość zakupu właśnie tego mikrokomputera.

W zeszłym roku pojawiło się kilka nowych, interesujących propozycji, czujemy się więc zobowiązani przynajmniej w sposób skrótowy je opisać. Dla oszczędności miejsca postaramy się, aby opisy konstrukcji były skondensowane, niemniej pozwalające zdobyć wstępne rozeznanie.

APPLE IIc

Wypełniając ubiegłoroczne przyrzeczenie swego szefa Steva Jobs'a („co rok nowy wyrób"), firma APPLE zaprezentowała na wystawie w San Francisco nowy mikrokomputer APPLE IIc (od ang. compact — zwarty). Rzeczywiście, IIc jest konstrukcją bardzo zwartą, ma bowiem wymiary 292×305×57 mm i waży ok. 3,5 kg.

Konstrukcja

APPLE IIc jest zbudowany na bazie mikroprocesora 65C02 (wersja CMOS), który w stosunku do 6502 ma dodatkowe instrukcje pozwalające tworzyć szybsze i bardziej zwarte oprogramowanie. Doświadczenia w projektowaniu i stosowaniu układów scalonych umożliwiły zmniejszenie ich liczby z 31 w IIe do 21 w IIc. Wśród nich znalazł się sterownik dysków elastycznych, zaprojektowany przez Steva Wozniaka (współzałożyciela firmy APPLE) i stąd nazywany IWM (Integrated Woz Machine).

Podstawowe cechy APPLE IIc:
— pamięć wewnętrzna: 128 KB RAM i 16 KB ROM (zawierająca Applesoft BASIC i Mouse Text)
— wbudowana stacja dyskietek 5,25" o pojemności 143 KB
— klawiatura 63-przyciskowa
— grafika 16-kolorowa o rozdzielczości 280×192 punkty lub 6-kolorowa o rozdzielczości 560×192 punkty

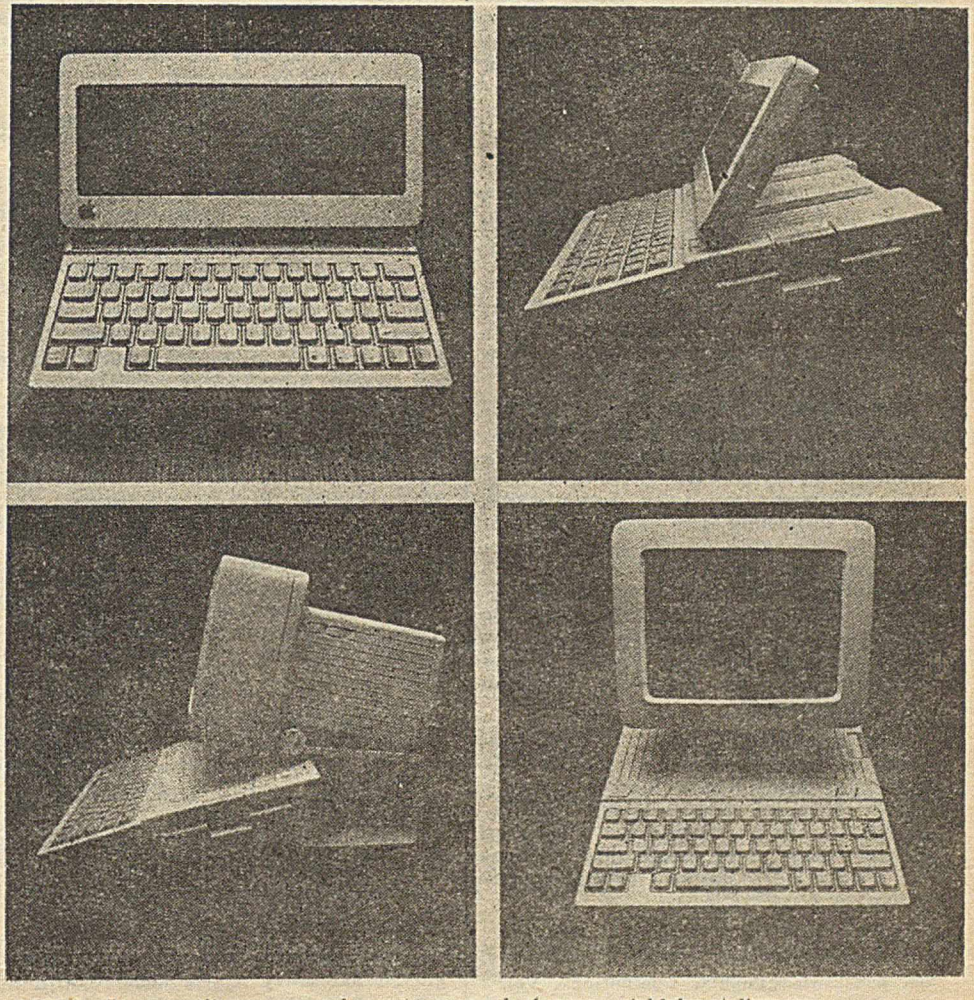
Do umieszczonych z tyłu obudowy gniazd w APPLE IIc można dołączyć myszkę lub manipulator drążkowy, modem (300 lub 1200 bodów), monitor kolorowy lub ekran ciekłokrystaliczny, monitor monochromatyczny, zewnętrzną stację dyskietek i drukarkę szeregową.

Klawiatura APPLE IIc jest bardziej spłaszczona niż w IIe (widać to na zdjęciach). Zastosowano też kilka nowych rozwiązań:
— możliwość przełączenia układu klawiatury ze standardowego QWERTY na układ Dvorak'a, który określa się jako przyszłościowy
— po prawidłowym wciśnięciu klawisza słychać cichy brzęk i daje się odczuć leciutkie drżenie klawisza (tzw. odpowiedź dotykowo-słuchowa)
— pod klawiszami zamontowano elastyczną podkładkę (ang. splash pad), która chroni wnętrze systemu przed zwarciem, jeśli na klawiaturę przypadkowo rozleje się jakiś płyn.

Użytkownik ma możliwość wyboru między 40- a 80-kolumnowym tekstem na ekranie. Wynika to z dwóch względów. Po pierwsze — wiele programów dostępnych na rynku napisanych jest z myślą o wyświetlaniu 40 znaków w linii. Po drugie — tekst 80-kolumnowy jest często niewyraźny na ekranie (konieczny jest monitor wysokiej klasy).

Najciekawszym elementem APPLE IIc jest ekran ciekłokrystaliczny, dostarczany przez japońską firmę SHARP. Pozwala on wyświetlać 24 wiersze 80-kolumnowego tekstu oraz przedstawiać grafikę o dużej rozdzielczości. Ekran ów znacznie podnosi walory APPLE IIc jako mikrokomputera przenośnego. Ponadto APPLE IIc może być zasilany napięciem 7...24 V (niebagatelna cecha na rynkach z dużą ilością samochodów i... różnorodnych akumulatorów).

W odróżnieniu od APPLE IIe, model IIc nie daje możliwości rozbudowy sprzętowej (nie ma dostępu do szyny systemowej), co nie pozwala np. na dołączenie pakietów z procesorem Z80 — nie można więc korzystać z bogatego oprogramowania napisanego dla CP/M. Jest to chyba najpoważniejsza wada nowego APPLE.



APPLE IIc z monitorem monochromatycznym i ekranem ciekłokrystalicznym

Oprogramowanie

APPLE IIc pracuje pod nowym systemem operacyjnym ProDOS (Professional Disk Operating System) kompatybilnym z poprzednikami: DOS3.3 (APPLE II) i SOS (Sophisticated Operating System — APPLE III). Główna różnica polega na włączeniu do ROM programu obsługi myszki (ang. Mouse Text), który generuje piktogramy (ang. icons) podobne do tych w mikrokomputerze MACINTOSH. Stąd ok. 10% programów napisanych dla APPLE IIe nie jest w pełni kompatybilnych z IIc i wymaga niewielkich przeróbek. Wiele z nowych programów używa już myszki w miejsce manipulatora drążkowego. Szerzej o oprogramowaniu APPLE pisano już w poprzednich numerach INFORMATYKI.

Nowy styl projektowania (ang. new design language)

Podobnie jak APPLE IIe, model IIc nie ma klawiszy funkcyjnych, by — jak stwierdzono — maksymalnie uprościć posługiwanie się mikrokomputerem. Temu celowi służy również obrazkowe przedstawienie funkcji każdego portu z tyłu obudowy.

APPLE IIc jest rezultatem szeroko zakrojonych badań rynku i zachowań użytkowników komputerów. Firma APPLE dostarczyła próbne egzemplarze specjalnie dobranym grupom użytkowników i zbierała opinie o pracy nowego mikrokomputera.

Okazało się, że użytkownicy małą wagę przywiązują do oprogramowania, uważając, że dobry sprzęt automatycznie gwarantuje sukces. Ponadto użytkownicy zbyt mało czasu poświęcają na dobre zapoznanie się z możliwościami sprzętu (grube książki-przewodniki!), stąd późniejsze kłopoty i — niejednokrotnie — zniecierpliwienie.

Wyciągając stąd wnioski, APPLE dostarcza 125-stronicowy przewodnik objaśniający sześć programów (umieszczonych na dyskietkach), które w sposób konwersacyjny uczą posługiwania się IIc. Programy owe zapoznają użytkownika zarówno z „zabawowym" wykorzystaniem APPLE IIc w grach komputerowych, jak i w poważniejszych przedsięwzięciach (przetwarzanie tekstów, korzystanie z bazy danych czy z programów kalkulacyjnych). Wprowadzają również elementy języków LOGO i BASIC.

Firma APPLE planuje, że 55% spośród wyprodukowanych modeli IIc znajdzie zastosowanie jako komputery domowe, reszta — w szkołach i w biurach. Nie ulega wątpliwości, że firma wiąże z nowym wyrobem duże nadzieje, przede wszystkim — w walce rynkowej z IBM PCjr.

opracował ADAM STAWOWY

na podstawie „Popular Computing", Vol. 3 No. 8

Publikujemy kolejny tekst, który jest wynikiem prac nad udoskonaleniem najtańszego mikrokomputera. Standardowy 1 KB pamięci to tak niewiele, że praktycznie każdy użytkownik po pewnym czasie kupuje rozszerzenie. Wbudowana wtedy w komputerze pamięć RAM marnuje się niewykorzystana. Podstawą prezentowanego w tekście pomysłu jest odzyskanie firmowego kilobajta RAM przy stosowaniu przystawki 16 KB. Można go wykorzystać do zdefiniowania odmiennego zestawu znaków (jak proponuje autor) lub do umieszczenia procedur w języku maszynowym.

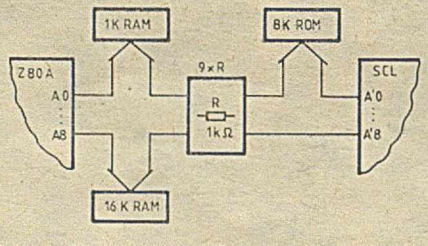
Opisana przeróbka wymaga ingerencji w budowę komputera, powinna więc być wykonywana przez osoby posiadające już pewne doświadczenie. Podjęte ryzyko można zmniejszyć starannie sprawdzając każdy wykonywany krok, natomiast postawa: „żeby tylko nie zepsuć" na pewno nie posunie nas do przodu.

Rozszerzenie możliwości graficznych ZX81 (16K)

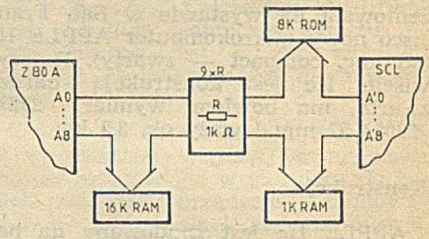
Jednym z głównych ograniczeń komputera ZX81 są jego małe możliwości graficzne. Ponieważ przystawki umożliwiające otrzymanie dużej rozdzielczości (256×192) dotychczas nie uległy w kraju szerszemu upowszechnieniu, proponuję rozwiązanie uproszczone.

Wykonując stosunkowo prostą przeróbkę połączeń wewnętrznych i stosując jeden dodatkowy układ scalony, możemy przesunąć tablicę znaków komputera z ROM-u do RAM-u. Pozwala to na deklarację nowego zestawu znaków, zależnie od potrzeb użytkownika mogą to być np. symbole elektroniczne czy litery innego alfabetu. Ponieważ nie ulega zmianie liczba możliwych znaków, jaką dysponuje komputer (64+64 negatywy), deklarując nowy znak musimy zrezygnować z jednego ze znaków starych — nie otrzymujemy więc możliwości stosowania tak uzyskanej wysokiej rozdzielczości w sposób nieograniczony.

Sposób kreślenia znaków w ZX81 polega na generacji przez SCL (ang. Sinclair Computer Logic) adresów kolejnych wierszy znaku zapisanego w tablicy 8×8 (8 bajtów na każdy znak) [2]. Adres początku tablicy znaków — 7680 pamiętany jest w rejestrze I mikroprocesora Z80. Generowane przez SCL adresy (A'_0—A'_8) nie mogą obciążać wyjść adresowych Z80 i z tego względu konstruktorzy ZX81 zastosowali oporniki maskujące (rys. 1). Oporniki te, dodatkowo, oddzielają wewnętrzną pamięć 1K RAM od SCL. Pamięć ta przy pracy z modułem 16K RAM jest blokowana i aby ją wykorzystać do kreślenia znaków, należy przejąć jej sterowanie (sygnałem RAMCS'/) i połączyć jej wejścia adresowe z adresami A'_0—A'_8 (rys. 2). Omawiane zmiany nie przekreślają możliwości używania komputera w wersji 1K, a dodatkowo umożliwiają wykorzystanie pamięci 1K przy pracy z modułem 16K (otrzymujemy 17K dostępnej pamięci RAM).

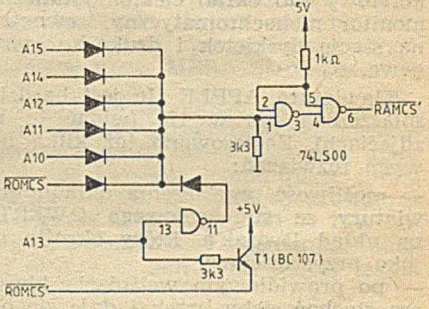


Rys. 1. Uproszczony schemat połączenia magistrali adresowej A_0—A_8 komputera ZX81 16K



Rys. 2. Schemat magistrali adresowej A_0—A_8 komputera po przeprowadzeniu opisywanych zmian

Przełączenie wejść adresowych A_0—A_8 RAM należy wykonać tak, aby nie zmieniły one sygnałów adresowych A_0—A_8 doprowadzonych do modułu 16K. Ze względu na dużą liczbę dodatkowych połączeń, jak też przecięć ścieżek na płytce drukowanej, przed włączeniem komputera należy kilkakrotnie sprawdzić prawidłowość wykonania tych zmian. Potwierdzeniem właściwego ich przeprowadzenia jest dobra praca komputera w wersji 1K. Do modułu 16K należy doprowadzić sygnał ROMCS/, rezygnując z jednego z sygnałów nie używanych przez ten moduł — np. Φ (zegar). Sygnał ten potrzebny jest układowi sterowania pamięci 1K przy pracy w wersji 17K z możliwością definiowania nowych znaków. Schemat układu przedstawia rysunek 3.



Rys. 3. Schemat układu sterowania pamięci 1K RAM

Układ ten należy umieścić wewnątrz modułu 16K RAM, co ze względu na małą liczbę użytych elementów nie powinno sprawić kłopotu. We wtyku modułu należy rozewrzeć połączenie końcówki RAMCS'/ z końcówką +5V i dołączyć do RAMCS'/ wyjście układu sterowania pamięci. Podczas pracy w wersji 17K, pamięć 1K umieszczona jest w obszarze od adresu 8192 do 9215, który został odblokowany przez tranzystor T1 (obszar 8-16K traktowany był przez procesor na równi z zajmowanym przez ROM obszarem 0-8K).

Po sprawdzeniu prawidłowości wykonania układu sterowania (nie należy stosować w nim bramek z serii standardowej), sprawdzamy jego działanie w połączeniu z komputerem (i z modułem 16K), np. za pomocą instrukcji POKE i PEEK. Aby przypisać tablicę znaków komputera obszarowi pamięci

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	1	0	0	1	0	0	36
0	0	1	1	0	1	0	0	52
1	1	1	1	1	1	1	1	255
0	0	1	1	0	1	0	0	52
0	0	1	0	0	1	0	0	36
0	0	0	0	0	0	0	0	0

Rys. 4. Przykład deklaracji nowego znaku

8-9K, należy zmienić, za pomocą krótkiego programu w języku wewnętrznym, wartość pamiętaną w rejestrze I z 30 na 32 lub 34 (nie należy używać 33 lub 35). W pierwszym przypadku początkiem tablicy znaków będzie adres 8192, a w drugim — 8704.

Położenie znaku w tablicy znaków określa jego kod. Znak o kodzie „0" zajmuje pierwsze osiem bajtów, następne osiem bajtów zajmuje znak o kodzie „1", itd. Jako przykład deklaracji nowego znaku, na rysunku 4 przedstawione są wartości kolejnych ośmiu adresów symbolu graficznego diody.

JACEK PSZONA
Warszawa

LITERATURA

[1] Dawidowski J.: Komputer osobisty ZX81. Informatyka, 5/1983
[2] Kaczmarczuk J.: Jeszcze o ZX81. Informatyka, 9/1983

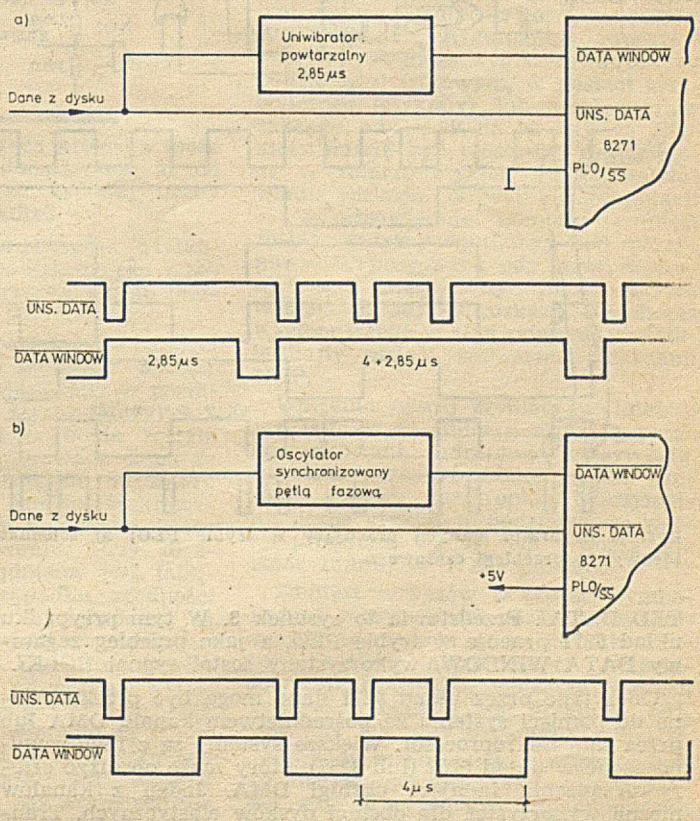
Drugą częścią artykułu o współpracy z polskimi napędami dysków elastycznych kończymy cykl publikacji o tym coraz popularniejszym typie pamięci masowej. Do tematyki dysków elastycznych zapewne jeszcze wrócimy — w chwili, gdy właściciele mikrokomputerów bardziej dojrzeją do tego problemu, również finansowo. Na razie skromne łamy mikroKLANU nie pozwalają na zaspokojenie wszystkich potrzeb. Zachęcamy jednak Czytelników do dzielenia się z nami swoją opinią o przydatności dotychczasowych publikacji. Będzie to bardzo pomocne przy dobieraniu do druku przyszłych materiałów.

Współpraca z napędem dysków elastycznych PL × 45D (2)

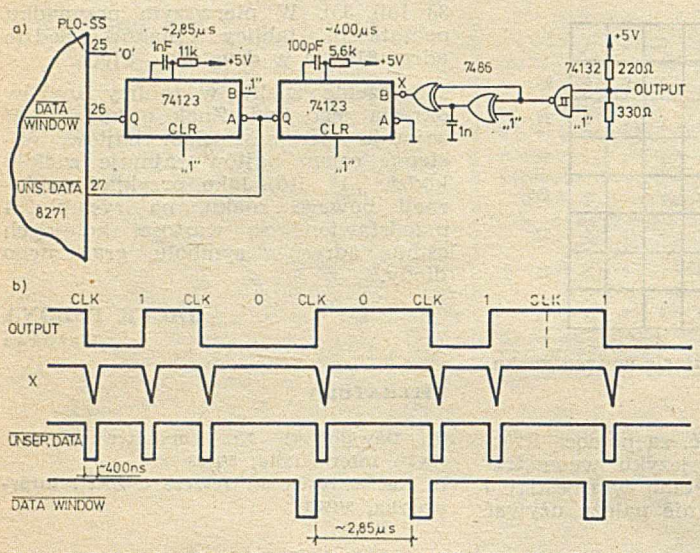
W układzie 8271 przewidziano dwa sposoby dekodowania danych, z sygnału odczytanego z dysku. Do wyboru sposobu służy wejście PLO-SS/. Na wejście UNSEPARATED DATA/ (wprowadzone są impulsy danych i zegara. Ich rozdzielenie możliwe jest dzięki wejściu DATA WINDOW/. Jeżeli na PLO-SS/ podany jest stan logiczny 0, to układ pracuje w trybie SINGLE SHOT. W tym przypadku sygnał DATA WINDOW/ wytwarzany jest z wykorzystaniem powtarzalnego uniwibratora, wyzwalanego od każdego impulsu danych i zegara (patrz rysunek 3 w poprzedniej części artykułu). Uniwibrator generuje impuls, jeżeli przez 2/3 okresu zegara (około 2,85 μs) na linii danych nie pojawi się sygnał. Oznacza to, że brak jest impulsu danych (dane=0) lub brak impulsu zegarowego.

Jeżeli na wejście PLO-SS/ podany jest stan logiczny 1, układ pracuje w trybie PHASE LOCKED OSCILATOR. W tym przypadku sygnał DATA WINDOW/ wytwarzany jest z wykorzystaniem generatora śledzącego częstotliwość sygnału odczytywanego z dysku. Jeżeli poziom wysoki wskazuje na impulsy zegarowe, to impulsy danych pojawiają się przy poziomie niskim. Faza przebiegu zegarowego nie jest istotna. Ważne jest natomiast, by sygnał generatora prawidłowo nadążał za zmianami częstotliwości odczytywanych danych. Oba tryby pracy przedstawiane są na rysunku 1.

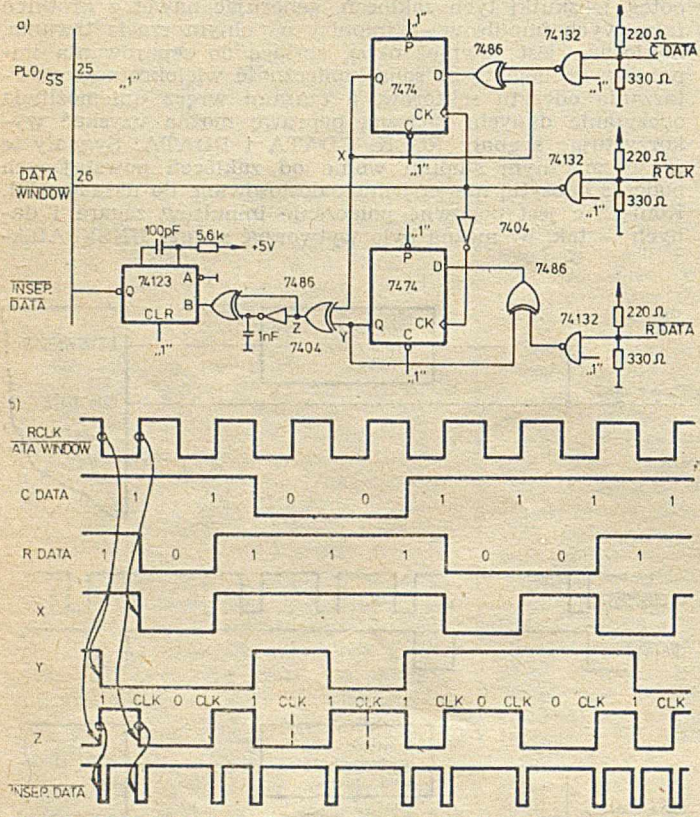
Sygnały odczytu danych z PLx45D można dołączyć do układu 8271 pracującego zarówno w trybie SS/, jak i PLO. W trybie SS/ sygnał UNSEPARATED DATA/ można wytworzyć z sygnału OUTPUT/. W przedstawionym na rysunku 2 układzie każde zbocze sygnału OUTPUT służy do wygenerowania ujemnego impulsu o czasie trwania ok. 400 ns. Impulsy te maskowane są przez kolejny uniwibrator, wytwarzający sygnał DATA WINDOW/. Ten prosty układ nie daje jednak w pełni zadowalających rezultatów. Sygnał OUTPUT/ jest mocno zakłócony, a przedstawiony układ potęguje skutki tych zakłóceń, generując nawet z krótkich fałszywych impulsów — impulsy o pełnym czasie trwania. Zakłócane jest również okno, służące do separowania impulsów. Pociąga to za sobą konieczność wielokrotnego powtarzania odczytu sektorów, a czasami wręcz uniemożliwia odczytanie danych. Znaczną poprawę można uzyskać wykorzystując sygnały RCLK, CDATA i RDATA. Sygnały te są w znacznym stopniu wolne od zakłóceń powstających podczas odczytu, nie są jednak dostosowane do układu 8271. Konieczne jest ponowne połączenie impulsów zegara i danych — tak, by można było wytworzyć sygnał UNSEPARA-



Rys. 1. Sposób separacji danych z sygnału modulowanego FM, a) z wykorzystaniem uniwibratora powtarzającego, b) z wykorzystaniem PLO



Rys. 2. Separator danych pracujący w trybie SS; a) schemat ideowy, b) przebiegi czasowe



Rys. 3. Separator danych pracujący w trybie PLO; a) schemat ideowy, b) przebiegi czasowe

TED DATA/. Przedstawia to rysunek 3. W tym przypadku układ 8271 pracuje w trybie PLO, a jako przebieg zegarowy DATA WINDOW/ wykorzystany został sygnał RCLK/.

Odczytane przez układ 8271 dane mogą być przekazywane do pamięci systemu za pośrednictwem kanału DMA lub przez sam mikroprocesor. Większe systemy są z reguły wyposażone w układ 8257 (lub 8237), który może obsłużyć cztery urządzenia żądające obsługi DMA. Jeden z kanałów można wykorzystać dla obsługi dysków elastycznych. Transmisja, z wykorzystaniem układu DMA znacznie upraszcza program obsługi dysków i sprowadza się do zainicjowania kierunku transmisji oraz adresu początku bufora danych.

Jeżeli w systemie nie ma układu DMA, procesor musi być całkowicie zaangażowany w przesyłanie danych. Czas, w jakim należy odebrać i przesłać dane, jest bardzo krótki, biorąc pod uwagę szybkość działania procesora 8080 oraz liczbę warunków, jaką musi on sprawdzić. Układ 8271 nie może oczekiwać na odebranie lub dostarczenie danych dłużej niż 31 µs. W tym czasie procesor musi sprawdzić waru-

```
                .
                .
                .
                LXI  H,ADDRESS          ; ADRES BUFORA
                MVI  C,RF  OR IRQ       ; MASKA KOŃCA OPERACJI
                MVI  B,CB  OR ND OR IRQ ; MASKA DANYCH
READ:           IN SREG                 ; POBIERZ STATUS   10T
                CMP B                   ; SĄ DANE ?         5T
                JNZ RESULT              ; TESTUJ DALEJ     10T
DATA:           IN DREG                 ; POBIERZ DANE     10T
                MOV M,A                 ; SCHOWAJ DANE      7T
                INX H                   ; NOWY ADRES        5T
                JMP READ                ; KONTYNUUJ        10T
RESULT:         CMP C                   ; KONIEC ?          5T
                JNZ READ                ; KONTYNUUJ        10T
                .
                .
                .
```

(Uwaga: CB=80H, RF=10H, IRQ=8, ND=4)

```
                .
                .
                .
                LXI D,ADDRES            ; ADRES BUFORA
                MVI H,PROGRAM           ; ADRES BAZOWY PROGRAMU
                MVI B,RF OR ND OR IRQ   ; MASKI
                .
                .
                .
READ:           IN  SREG                ; POBIERZ STATUS
                ANA B                   ; WYTNIJ BITY
                MOV L,A                 ; MŁODSZY BAJT BAZY PROGRAMU
                PCHL                    ; SKOK DO PROCEDURY
                .
                .
                .
DATA:           IN  DREG                ; POBIERZ DANĄ
                STAX D                  ; SCHOWAJ DANĄ
                INX D                   ; NOWY ADRES
                JMP READ                ; KONTYNUUJ
                .
                .
                .
RESULT: IN RREG                         ; POBIERZ WYNIK
                .
                .
                .
```

nek gotowości danych, warunek zakończenia operacji, odebrać dane i zapisać je do pamięci. Powyżej przedstawiono fragment programu w języku ASSEMBLER 8080, realizujący wymienione czynności oraz czas jego wykonania.

Najdłuższy okres oczekiwania na dane występuje w przypadku wykonania następującego ciągu instrukcji: CMP (5T), JNZ (10T), CMP (5T), JNZ (10T), IN (10T), CMP (5T), JNZ (10T) oraz IN (10T) — w sumie 65 cykli zegarowych T. Przy największej częstotliwości zegara, z jaką może pracować mikroprocesor 8080 (T=0,48 μs), daje to czas 31,2 μs. Najdłuższy czas realizacji zapisu jest jeszcze większy — ze względu na konieczność pobrania danych z pamięci przed przesłaniem ich do układu.

Przedstawiony przykład pokazuje jak krytyczny jest czas, w którym mikroprocesor musi obsłużyć układ 8271 podczas transmisji danych. Poniżej przedstawiony zostanie zarys programu rozwiązującego problem (druga część wydruku).

Program wymaga odpowiedniego ustawienia poszczególnych fragmentów w pamięci, przy czym niektóre jego części muszą być powtórzone. W tym przypadku za czas wykonania trzeba zapłacić obszarem zajmowanym w pamięci.

Przy tak napisanym programie udało się uzyskać dla najgorszego przypadku przy odczycie danych czas odpowiadający 50 cyklom zegarowym, a przy zapisie — 57 cyklom zegarowym.

Program obsługi dysków elastycznych powinien być dostosowany do potrzeb przewidywanego systemu operacyjnego. Systemy operacyjne z reguły dokonują transmisji pomiędzy pamięcią a dyskiem, poprzez własny bufor danych o długości jednego sektora. Pojedyncza operacja dyskowa dotyczy z reguły tylko jednego sektora, a po jej zakończeniu przygotowywana jest kolejna zawartość bufora. Takie działanie powoduje znaczne wydłużenie czasu współpracy z dyskami, gdyż podczas przygotowywania bufora dyskietka zdąży obrócić się na tyle, że kolejny sektor minie już głowicę odczytującą. Dostęp do tego sektora możliwy będzie dopiero po pełnym obrocie dysku (ok. 1/6 s). Przyspieszenie operacji dyskowych można uzyskać przez fizyczną lub logiczną zmianę numeracji sektorów, czyli przez sformatowanie ścieżki w taki sposób, by sektory nie występowały w kolejności swoich numerów (ale np. co drugi lub co trzeci) lub przez nie zapisywanie zbioru w kolejnych sektorach.

Znacznie lepsze wyniki można byłoby uzyskać dokonując jednoczesnej operacji na kilku sektorach, na co pozwala układ 8271, lub przez zrównoleglenie przesyłania danych kanałem DMA i przygotowywanie drugiego bufora danych — tak, by kolejną operację można było rozpocząć natychmiast po zakończeniu poprzedniej. Wybór rozwiązania zależy jednak od specyfiki konkretnego systemu operacyjnego.

ADAM PLUTA
Warszawa

Nazbyt może skrótowy opis systemu DS-80 MONITOR zamieszczamy z nadzieją, że mimo wszystko pozwoli się on zorientować w podstawowych możliwościach systemu. Koncepcja zastosowana w DS-80 znacznie odbiega od przyjętej w popularnych systemach operacyjnych; jest ona — jak się wydaje — szczególnie atrakcyjna dla mikrokomputerów nie posiadających szybkich pamięci masowych, czyli — korzystających z magnetofonu.

Jeszcze raz powtarzamy zasadę: nie marnujmy czasu na robienie tego, co już zostało zrobione! Opisane oprogramowanie praktycznie każdy może uzyskać na dogodnych zasadach (nawet w wersji źródłowej). Oczywiście, odbiorcy profesjonalni muszą liczyć się z pewnymi nakładami. Pod tekstem podajemy numer telefonu autorów.

DS-80 MONITOR

Jako podstawowe urządzenia peryferyjne systemu, w którym można wykorzystać program DS-80 MONITOR stosowane są: monitor z wejściem wizyjnym, klawiatura alfanumeryczna, magnetofon kasetowy.

Program mieści się w pamięci EPROM o pojemności 4 KB i korzysta z 4 KB pamięci RAM przeznaczonych na pamięć obrazu i obszar roboczy.

DS-80 MONITOR umożliwia realizowanie następujących operacji:

- odczytywanie zdisasemblowanej zawartości pamięci w standardowej formie edycyjnej (adres — kod maszynowy — etykieta — mnemonik)
- wprowadzanie programów w języku ASSEMBLER z możliwością stosowania etykiet, adresowania bezwzględnego, względnego i symbolicznego
- wpisywanie do pamięci liczb w kodzie szesnastkowym
- relokowanie, rozsuwanie (w celu dopisania dodatkowych rozkazów) lub usuwanie zbędnych fragmentów programu z jednoczesną korekcją adresów
- uruchamianie programów użytkowych od dowolnego adresu
- realizację pracy krokowej (z możliwością modyfikacji zawartości rejestrów procesora); praca krokowa może być prowadzona automatycznie aż do żądanego adresu lub zadanej liczby przejść przez taki adres
- wprowadzanie programów użytkowych z magnetofonu kasetowego i zapis zawartości pamięci na kasetę (modulacja amplitudy; 600 bodów).

Odczytywanie zawartości pamięci rozpoczyna się od zadanego adresu. Adres może być deklarowany w postaci absolutnej (kod szesnastkowy) lub symbolicznej (etykieta), o ile została ona wcześniej zadeklarowana (i co za tym idzie — wprowadzona do zbioru etykiet). Zawartość pamięci traktowana jest jako program wynikowy, który podlega disasemblowaniu. Przy disasemblowaniu uwzględniana jest tablica etykiet i w przypadku zgodności adresów — etykiety wprowadzane są w pole etykiety lub jako operandy instrukcji 3-bajtowych. Zawartość pamięci wyświetlana jest na ekranie monitora w wierszach po jednej instrukcji. Kolejne wiersze wyświetlane są u dołu ekranu po przepisaniu w górę poprzedzających wierszy. Na ekranie mieści się 26 wierszy zawierających nie więcej niż 64 znaki. Wyprowadzanie odbywa się tak długo, jak długo wciśnięty jest klawisz SPACE. Może być ono wznowione po ponownym wciśnięciu tego klawisza. Zadeklarowanie nowego adresu powoduje wymazanie poprzedniej zawartości ekranu.

Wprowadzanie programów lub danych może być realizowane w postaci symbolicznej lub w kodzie szesnastkowym. Przed rozpoczęciem wpisywania należy zadeklarować adres początkowy. Rozkazy w postaci symbolicznej wpisuje się zgodnie z normami języka ASSEMBLER. Argumenty rozkazów 3-bajtowych mogą być wprowadzone w kodzie szesnastkowym, w postaci symbolicznej (etykiety) lub względnej w zakresie od — 99 do 99 bajtów. Etykiety muszą być wcześniej umieszczone w zbiorze etykiet. Po skompletowaniu rozkazu, w wyniku asemblacji — w odpowiednie komórki pamięci zostaje wpisany wynikowy kod maszynowy. W przypadku gdy nowo wpisany kod maszynowy nie pokrywa wszystkich bajtów rozkazu dotychczas występującego w tym miejscu, pozostałe bajty zostają zastąpione rozkazami NOP.

Wpisanie nowej etykiety w miejsce już istniejącej zmienia etykietę przyporządkowaną adresowi wpisywania. Etykieta może być usunięta przez wpisanie samego dwukropka. Maksymalna liczba etykiet w podstawowej wersji programu wynosi 145. Tworzą one zbiór zajmujący 1 KB pamięci.

Zamiast rozkazów w postaci symbolicznej, można wprowadzać liczby szesnastkowe złożone z dwóch lub czterech znaków zakończonych przecinkiem. W przypadku wpisania czterech znaków zmieniana jest kolejność bajtów, tak jak przy pseudorozkazie DW.

Rozkazy w postaci symbolicznej mogą być wpisywane tylko w przypadku, gdy adres wpisywania odpowiada pierwszemu bajtowi rozkazu dotychczas znajdującego się w pamięci. Liczby szesnastkowe mogą być wpisywane w dowolne miejsce.

W procesie wpisywania kontrolowana jest formalna poprawność zapisu; w przypadku wystąpienia jakichkolwiek błędów — są one sygnalizowane napisem ERROR, a błędne zlecenie nie jest wykonywane.

Zastosowany sposób wprowadzania daje umieszczenie w pamięci mikrokomputera absolutnego programu wynikowego, który może być uruchomiony natychmiast po zakończeniu wpisywania.

Programy mogą być przechowywane na kasetach magnetofonowych. Przed wyprowadzeniem programu należy zadeklarować jego adres początkowy i końcowy. Razem z programem na taśmę zapisywana jest także zawartość zbioru etykiet.

Uzupełnieniem programu operacyjnego są wprowadzane z taśmy magnetofonowej procedury:

— drukowania zawartości pamięci w standardowej formie edycyjnej za pomocą drukarki DZM 180 KSR.

— drukowania zawartości pamięci z możliwością umieszczenia komentarza — programowania pamięci EPROM typu 2708 i 2716.

Opracowano także rozszerzoną wersję programu o objętości 5 KB, w której przewidziano możliwość specjalnego wyprowadzania tabel oznaczonych etykietami rozpoczynającymi się od liter TB (tablica danych 1-bajtowych), TW (tablica danych 2-bajtowych) i TA (tablica tekstów w kodzie ASCII). Zwiększono także liczbę etykiet do 255, dołączono program drukowania oraz wprowadzono możliwość sprawdzania poprawności zapisu programu na taśmie magnetofonowej. Podobną wersję programu operacyjnego opracowano także dla mikroprocesora Z-80 (Z-80.S MONITOR). Ze względu na bardziej złożony ASSEMBLER, program ten ma objętość 6 KB.

Jak wynika z doświadczeń, omówione programy operacyjne są bardzo dobrym narzędziem programowania w języku ASSEMBLER oraz uruchamiania i testowania programów użytkowych. Ich modularna struktura umożliwia wykorzystywanie zawartych w nich podprogramów przez programy użytkowe. Dotyczy to obsługi operacji we-wy, konwersji kodów, pętli opóźniających, obsługi transmisji DMA i innych tego typu operacji. Program jest szybki: przesuwanie bloku o objętości 4 KB przy maksymalnej liczbie etykiet (255) wraz ze zmianą adresów trwa nie dłużej niż kilka sekund. Praca z systemem jest znacznie szybsza i wygodniejsza niż przy użyciu firmowego EDITORA i ASSEMBLERA.

Mankamentem systemu jest brak możliwości wprowadzania komentarzy podczas pisania programu; można je wprowadzać jedynie w trakcie końcowej dokumentacji programu.

GRZEGORZ STĘPIEŃ
MIROSŁAW DOLEŻYCH
JERZY ORKISZEWSKI
Politechnika Warszawska, IBSPiE
tel. 49-98-71 w. 538

Konsekwentnie wprowadzana reforma spowodowała, że wiele rzeczy (często podstawowych) ma zbyt wysoką cenę jak na możliwości przeciętnego obywatela. O mikrokomputerach trudno jednak powiedzieć, że są drogie. Są po prostu poza możliwościami finansowymi znakomitej części społeczeństwa.

Propozycja kupowania używanych komputerów oczywiście nie rozwiązuje tego problemu. Staramy się jednak nie zaprzepaścić żadnej, nawet najdrobniejszej, szansy upowszechniania mikroinformatyki.

Komputery z drugiej ręki

W poszukiwaniu taniego (a więc odpowiadającego naszym zasobom gotówki) mikrokomputera bezsprzecznie należy rozważyć możliwość zakupu używanego egzemplarza. Należy jednak przedtem znaleźć odpowiedź na kilka podstawowych pytań:

— skąd pochodzi używany komputer

— czy będzie pracował jako całość czy też należy go raczej potraktować jako źródło części zamiennych

— czy dostępne są instrukcje obsługi, czy też wszystko trzeba będzie wynajdywać metodą Sherlocka Holmesa

— czy jego wartość odpowiada cenie.

Gwałtowny rozwój mikroinformatyki znajduje swoje odbicie w stosunkowo częstej wymianie sprzętu przez użytkowników. Szczególnie przez tych profesjonalnych. Dla nadążenia za postępem w tej dziedzinie zalecana jest wymiana komputera mniej więcej co dwa lata (!). Wiele instytucji próbuje jednak pogodzić to zalecenie z zasadami ekonomiki czyli wycisnąć z komputera ile tylko się da — dopóki jego eksploatacja jest opłacalna (szczególnie te mniej zamożne — na przykład uczelnie). Z kolei niektóre komputery spełniają głównie rolę eksponatów i można powiedzieć, że są praktycznie nowe. Tak więc w zależności od rodowodu używany komputer może reprezentować różną wartość.

Tańsze mikrokomputery, przeznaczone w zasadzie do domowego użytku przez prywatnych odbiorców (na przykład: VC 20, ORIC, ZX SPECTRUM, C 64, ATARI), różnią się między sobą niezawodnością konstrukcji lecz generalnie rzecz biorąc nie są one produkowane z myślą o eksploatacji przez długie lata. Co gorsza, wiele modeli jest wprowadzanych na rynek z dużym pośpiechem, bez dokładnego przetestowania konstrukcji i dopiero druga albo nawet trzecia wersja jest wolna od błędów. Czasem szwankuje współpraca z urządzeniami peryferyjnymi, innym razem okazuje się, że nie dopracowano oprogramowania firmowego, a kłopoty z przegrzewaniem się mikrokomputerów należą prawie do reguły (np. ZX SPECTRUM, APPLE II).

Jeżeli dostępny jest komputer z poprawionej wersji to można oczekiwać, że przez parę ładnych lat będzie funkcjonował praktycznie niezawodnie. Mogą jednak wystąpić trudności z rozpoznaniem wersji z którą mamy do czynienia. Tylko sporadycznie, jak na przykład w przypadku komputera OSBORNE, producent wprowadza jakiś element rozróżniający (tu zmieniono kolor obudowy). Przykładowo, w ZX SPECTRUM trzeba by zdjąć obudowę aby przekonać się, że na płytce drukowanej umieszczono napis „issue 3".

Z dobrym przybliżeniem można jednak założyć, że nawet w przypadku trafienia na nie dopracowaną wersję wszystkie układy scalone będą sprawne i jeżeli znamy istotę wprowadzonych przez producenta poprawek, to często możemy dokonać zmian we własnym zakresie.

Po kilkuletniej niezawodnej eksploatacji wzrasta jednak prawdopodobieństwo pojawienia się usterek. Po pierwsze mogą wystąpić mikropęknięcia wokół wyprowadzeń układów scalonych, umożliwiające wniknięcie pary wodnej i rozpoczęcie procesów korozji. Zdarzają się też pęknięcia ścieżek drukowanych lub też uszkodzenia układów scalonych, wywołane przez zwężenie się wewnętrznych mikropołączeń w wyniku migracji nośników elektrycznych.

W przypadku sprzętu profesjonalnego szacuje się, że po ok. siedmiu latach usterki mogą nasilić się do tego stopnia, że dalsza eksploatacja przestaje być opłacalna.

Najbardziej podatnymi na zużycie częściami mikrokomputera są wszelkie elementy elektromechaniczne, a więc klawiatura, napędy dysków elastycznych, drukarki lub magnetofony. W przypadku monitorów ekranowych dość częstym zjawiskiem jest wypalenie lampy kineskopowej.

Jeżeli podejmiemy decyzję o zakupie używanego komputera to istnieją dwie możliwości:

— zakupić odpowiednie pismo fachowe lub hobbystyczne i wśród prywatnych ogłoszeń znaleźć interesującą nas ofertę

— wybrać się do sklepu prowadzącego skup i sprzedaż używanego sprzętu komputerowego.

W pierwszym przypadku występuje dosyć duże ryzyko związane z tak zwanymi wadami ukrytymi. Niektóre uszkodzenia dają o sobie znać dopiero po pewnym czasie, inne tylko w specyficznych sytuacjach. Trzeba więc mieć trochę doświadczenia i przysłowiowy łut szczęścia.

W przypadku sklepu występuje trochę inna sytuacja: przed odsprzedażą sprzęt poddawany jest testom, gdyż zazwyczaj udzielana jest miesięczna gwarancja. Nie eliminuje to oczywiście możliwości wystąpienia uszkodzeń, ale jednak poważnie ją zmniejsza. Oczywiście za pośrednictwo i testowanie pobierana jest opłata i wynosi ona około 30%. Do tego Brytyjczycy muszą jeszcze doliczyć 15% podatku. Przybysze z Polski po otrzymaniu na granicy potwierdzenia wywozu, mogą ze stratą w wielkości opłat manipulacyjnych odzyskać zapłaconą kwotę podatku (tzw. VAT — ang. Value Added Tax).

A oto przykładowe ceny (w funtach) używanych mikrokomputerów (łącznie z podatkiem) — listopad 1984:

BBC model B	— 325
ACORN ATOM	— 40
COMMODORE C64	— 139
EPSON HX-20	— 325
DRAGON	— 65
NEWBRAIN A	— 103
SPECTRUM	— 99
JUPITER ACE	— 15
LYNX	— 115
NEWBRAIN AD	— 110
PC-1500	— 65
z drukarką i magnetofonem	75

Warto zwrócić uwagę, że na przykład całkiem niezły mikrokomputer ACORN ATOM po prostu wyszedł z mody i w porównaniu do innych oferowany jest prawie za bezcen. Wadą JUPITERA ACE jest fakt, że zamiast interpretera języka BASIC ma on zaimplementowany FORTH. Jednak ci, co potrafią programować w tym języku na pewno będą odmiennego zdania i mimo niewielkiej pamięci RAM (16 KB) zdecydują się na zapłacenie tej prawie symbolicznej ceny.

Oprócz komputerów, w sklepach z używanym sprzętem często można nabyć naprawdę okazyjnie urządzenia peryferyjne. Oto przykłady. Ośmiocalowy dysk Winchester o pojemności 19 MB — praktycznie nowy — oferowany był przez jeden z londyńskich sklepów za... 280 funtów. Drukarka firmy IBM GOLFBALL sprzedana była za 80 funtów.

Niektóre urządzenia niezłej klasy sprzedawane są po niskiej cenie z absurdalnych, jak na nasze wyobrażenia, powodów. Na przykład, ploter bębnowy CALCOMP o dokładności 1/100 cala — i to nie tylko na szerokości papieru, ale i wzdłuż całej rolki o długości 36 m — można było kupić za 165 funtów. Jednak linie interfejsu tego plotera zamiast standardowych poziomów TTL wymagają sterowania napięciem rzędu 10V.

Można oczekiwać, że podane informacje wzbudzą wiele emocji wśród Czytelników, wybierających się właśnie na Zachód. Niestety redakcja dysponuje tylko telefonami sklepów z używanym sprzętem komputerowym na terenie Wielkiej Brytanii (a i to zapewne nie wszystkich). Osoby, które mają podobne informacje z innych krajów prosimy o udostępnienie swoich doświadczeń (za naszym pośrednictwem) pozostałym Czytelnikom.

AJP

W Akademii mikroKLANU nie staramy się uczyć programowania w języku BASIC. Znaczenie poszczególnych instrukcji tego języka można znaleźć w dołączanej do każdego mikrokomputera instrukcji obsługi. Na dodatek istnieje niemal tyle odmian tego języka ile typów mikrokomputerów.

Akademię mikroKLANU adresujemy do tych, którzy mają już za sobą pierwsze poprawnie działające programy. Chcemy im pokazać na czym polega elegancja w pisaniu programów. Bardziej doświadczonych Czytelników zapraszamy do współpracy w Akademii.

Akademia mikroKLANU (3)

Koniunkcja

W tym odcinku zajmiemy się zasadniczą częścią programu czyli tą, w której wykonywane są manipulacje na danych. Pamiętamy już, że program powinien być nie tylko łatwy w obsłudze lecz także zrozumiały dla innych — jest to tzw. **reguła kumpla**. Poniższy program, aczkolwiek może zostać zrealizowany przez większość komputerów nie wygląda zbyt zachęcająco:

```
LIST

10 INPUT"DODATNIA LICZBA CALKOWITA=";N
20 M=N
30 IFN=2THENGOTO80
40 N=N/2
50 IFINT(N)=NTHENGOTO30
60 N=N*6+1
70 GOTO30
80 PRINT"LICZBA ";M;" SPELNIA ZASADE ULAMA!"
90 END
READY.
```

A przecież opisane w poprzednich odcinkach reguły tworzenia wejścia i wyjścia były tu skrupulatnie przestrzegane! Autor tego programu posunął się za daleko w oszczędności pamięci komputera i przez to zupełnie zatracił czytelność programu. Niestety wiele osób uważa, że pisanie oszczędnych programów polega tylko na oszczędzaniu pamięci komputera. Szkoda, że nie potrafią oszczędzać własnego i cudzego czasu... W przykładzie brak też jakichkolwiek komentarzy co program robi. Zastanówmy się, jakie należy wprowadzić zmiany, by program stał się w miarę czytelny.

Po pierwsze należy pisać z odstępem między instrukcjami — (niektóre interpretery języka BASIC automatycznie wstawiają spacje między instrukcje).

Po drugie komentarze powinny towarzyszyć większości operacji (oczywiście nie trzeba komentować faktu dzielenia przez dwa lecz raczej podać **kiedy** program dzieli przez dwa, jeżeli nie jest to oczywiste).

Wygodnie jest umieszczać komentarze w linijkach z numerami nie będącymi wielokrotnością 10 (za to instrukcje tylko w liniach o numerach będących wielokrotnością 10). W ten sposób ktoś, kto będzie przepisywał nasz program i ewentualnie przystosowywał go dla innego komputera, może łatwo pominąć niektóre komentarze uzyskując dodatkowe miejsce dla dokonania poprawek wynikających ze specyfiki konkretnego komputera.

Dobrze jest też zamieszczać w programie instrukcje wyjścia w środku pętli. W przykładowym programie już dla N równego 31, czas wykonania programu jest dość znaczny — warto więc sygnalizować użytkownikowi, że program liczy. Przedłuża to nieco czas realizacji programu, ale daje pewność, że program działa poprawnie. Jest to szczególnie istotne przy testowaniu poprawnej pracy programu. Przykładowy program można więc uzupełnić:

```
75 PRINT N;
95 PRINT N;
```

Gdy przekonamy się, że program działa poprawnie, można linie 75 i 95 opuścić.

Warto umieścić w programie kilka instrukcji zbierających dane statystyczne (np. o liczbie cykli, które zostały wykonane i największej liczbie, do której doszliśmy w naszej procedurze.

Warto też, po dokonaniu wszystkich poprawek, przenumerować linie programu — znowu zyskamy instrukcje tylko w linijkach mających numery będące wielokrotnością dziesięciu.

Tak więc w rezultacie przykładowy program ma następującą postać:

```
LIST

5 REM PROGRAM SPRAWDZA ZASADE ULAMA:JEZELI DODATNIA
6 REM LICZBA CALKOWITA JEST PARZYSTA TO NALEZY PODZIELIC
7 REM JA PRZEZ 2,GDY JEST NIEPARZYSTA TO POMNOZYC PRZEZ 3
8 REM I DODAC 1 A CALA PROCEDURA POWTARZYC AZ DO OTRZYMANIA
9 REM LICZBY 1 W SKONCZONEJ ILOSCI KROKOW.
10 INPUT "DODATNIA LICZBA CALKOWITA=";N
20 IF N<1 OR INT(N)<>N THEN GOTO 10
30 M=N
40 I=0
41 REM I JEST LICZNIKIEM ILOSCI CYKLI
50 X=1
51 REM X JEST NAJWIEKSZA OSIAGNIETA LICZBA
60 I=I+1
70 IF N=2 THEN GOTO 130
71 REM JEZELI N=2 TO  NASTEPNYM KROKU BYLOBY ROWNE 1
80 N=N/2
90 IF INT(N)=N THEN GOTO 60
91 REM POPRZEDNIE N BYLO PARZYSTE
100 N=N*6+1
101 REM POPRZEDNIE N BYLO NIEPARZYSTE I ZOSTALO JUZ PODZIELONE PRZEZ 2
110 IF N>X THEN X=N
120 GOTO 60
130 PRINT "LICZBA ";M;" SPELNIA ZASADE ULAMA!"
140 PRINT "ILOSC CYKLI DO OSIAGNIECIA 1 WYNOSI ";I
150 PRINT "NAJWIEKSZA OSIAGNIETA LICZBA WYNOSI ";X
160 END
READY.


RUN
DODATNIA LICZBA CALKOWITA=? 31
LICZBA  31  SPELNIA ZASADE ULAMA!
ILOSC CYKLI DO OSIAGNIECIA 1 WYNOSI  106
NAJWIEKSZA OSIAGNIETA LICZBA WYNOSI  9232

READY.
```

W tym programie niestety nie udało się nam uniknąć stosowania instrukcji bezwarunkowej **GOTO** (linia 120). Teoretycy informatyki (patrz np. N. Wirth „Wstęp do programowania systematycznego". WNT, 1978) uważają, że dobry program nie powinien zawierać takich instrukcji. W języku BASIC podprogramy są też wywoływane przez numer linii, a więc strukturalnizacja programu niewiele polepsza jego zrozumiałość. Ale o tym będzie mowa w następnych odcinkach!

J. TATARKIEWICZ

Tak zwana zasada Ulama wcale nie została podana przez tego wybitnego matematyka polskiego pochodzenia! On sam mówi o niej: „Jest ona taka piękna, że chciałbym być tym co ją wymyślił!" Z dostępnej literatury wynika, że zasada ta występuje pod różnymi nazwami (algorytm z Syrakuz, problem Collatza-Kakutani i in.). Do dzisiaj nie tylko nie podano dowodu, iż zasada jest poprawna dla wszystkich liczb całkowitych, ale także nie ma dowodu, że koniunkcji tej nie da się udowodnić. Dr Bryan Thwaites z Westfield College w Anglii ufundował nagrodę 1000 funtów dla każdego kto przedstawi poprawny dowód zasady lub uzasadnienie niemożliwości takiego dowodu. Wiadomo, że przetestowano już liczy aż do około 2^{50} i zawsze zasada była prawdziwa. A może ktoś z Czytelników INFORMATYKI zdobędzie nagrodę?

Ceny sprzętu komputerowego w Anglii, marzec 1985 (w funtach — bez podatku V.A.T.)

COMMODORE C64 — 166.—; COMMODORE C16 (+ magnetofon) — 100.—; COMMODORE PLUS 4 — 200.—; COMMODORE 1541 (stacja dysków) — 160.—; COMMODORE 801 (drukarka) — 130.—; COMMODORE 802 (drukarka) — 261.—; ACORN ELECTRON — 110.—; BBC (model B) — 290.—; SINCLAIR QL (128KB) — 335.—; SPECTRUM 48K PLUS 107.—; MICRODRIVE + INTERFACE1 — 83.—.
Obudowa zamieniająca SPECTRUM w SPECTRUM PLUS — 20.—
AMSTRAD CPC 464 (+ monitor monochromatyczny) — 209.—
AMSTRAD CPC 464 (+ monitor kolorowy) — 301.—
SERIA MSX:
TOSHIBA HX10 64K — 190.—; SANYO MCR10 64K — 235.—; SONY HB75B 64K —245.—; JVC HC7GB 64K — 226.—.
DRUKARKI:
SEIKOSHA GP500A — 113.—; SEIKOSHA GP700A Colour — 244.—; SHINWA CTA80 — 200.—; EPSON RX80.

Ceny podzespołów w RFN, luty 1985 (w markach z podatkiem MWST)

PAMIĘCI:
2114 — 7.95, 4116 (200) — 3.95, 4164 (250) — 13.50, 41256 (150) — 79.—, 2708 — 8.70, 2716 (450) — 13.80, 2732 (350) — 18.90, 2732 (450) — 15.90, 2764 (250) — 21.—, 27256 (250) — 195.—,
MIKROPROCESORY:
Z80A — 6.50, Z80B — 15.—, 6502P — 13.90, 6502A — 15.25, 6800 — 10.30, 6802 — 9.95, 6809 — 22.90, 68000 (8MHz) — 82.—,
STEROWNIKI CRT:
6845 — 16.50, EF 9365(96)7 — 85.—, UPD 7220 — 79.—
STEROWNIKI Nnapędów DYSKOWYCH:
1771 — 39.—, 1791 — 27.50, 1797 — 27.50, 2793 — 65.—, 2797 — 65.—, 1691 — 26.50, 9229B — 89.—

- W Tarnowskich Górach od października 1984 rozpoczął działalność Klub Mikrokomputerowy, który zadomowił się w lokalu Klubu Technika w dzielnicy Strzybnica. Kontakt: Piotr Drewniak (prezes), ul. W. Wasilewskiej 4/5; 42-609 Tarnowskie Góry tel. d. 85-83-01, tel. praca 85-22-41 w. 585, (kierunkowy jak na Katowice)
- Przy ZR Kasprzaka w Warszawie rozpoczął działalność klub mikrokomputerowy „Mikros". Informacje: Waldemar Kufel, tel. p. 32-99-27 lub 32-16-21
- Zapowiadane broszury mikroKLANU (kilka jest już gotowych!) zapewne ukażą się... nie wcześniej jak w roku 1986. Naiwnie sądziliśmy, że broszura drukuje się najwyżej kilka miesięcy.

Przedsiębiorstwo Polonijno-Zagraniczne ITM

31-546 Kraków, ul. Mogilska 80

poleca

sprzęt komputerowy z oprogramowaniem

- komputer osobisty MERITUM I (16, 32 lub 48K RAM)
- drukarka Seikosha typ GP-500A
- generator w.cz. umożliwiający podłączenie Meritum do dowolnego odbiornika TV
- biblioteka programów matematycznych i statystycznych
- oprogramowanie Asembler/Edytor dla Z80 oraz INTEL 8035
- oprogramowanie specjalne na zamówienie użytkownika

EO/391/K/85

prowadzi
ANDRZEJ J. PIOTROWSKI
tel. dom.: 48-22-85

dokończenie ze str. 12

Przedmiotem badania była także ocena przez respondentów własnych kwalifikacji. Za wystarczające uznało je 24,1% respondentów; za wystarczające, ale przy systematycznym samokształceniu — 45,9%; zaś 28,6% respondentów uznało, że aby posiadać kwalifikacje zawodowe należy ukończyć kurs specjalistyczny. Nieliczni respondenci stwierdzili, że posiadane kwalifikacje pozostają bez związku z wykonywaną pracą.

Z przedstawionych danych wynika, że wykształcenie dopiero w powiązaniu z kursami, z systematycznym samokształceniem może dać niezbędne kwalifikacje zawodowe.

Niezwykle interesujące było stwierdzenie w jakim stopniu praca w obsłudze technicznej spełnia oczekiwania zatrudnionego personelu; jakie są przyczyny obserwowanej w kraju dużej fluktuacji w tej grupie pracowników. Odpowiedzi na tak postawione pytanie były następujące:

— pełne zadowolenie z pracy w obsłudze technicznej ma 4,5% respondentów
— w zasadzie praca ta daje zadowolenie — 40,9%
— daje tylko pewną satysfakcję — 33,6%
— początkowo praca dawała zadowolenie, a teraz nie — 18,6%
— praca ta nie dawała i nie daje zadowolenia — 2,3%.

Widać jak znikomy procent personelu obsługi technicznej w pełni identyfikuje się z wykonywanym zawodem.

Chęć ewentualnej zmiany pracy deklaruje 59,1% respondentów:

— na inną pracę, nie związaną z techniką komputerową — 22,7%
— na pracę w innej instytucji o charakterze informatycznym — 20,5%
— na inną w swojej instytucji (dziale) — 19,9%

Jako podstawowe motywy skłaniające do zmiany pracy wymieniono (po dwa jednocześnie):

— otrzymanie wyższej płacy — 61,8%
— lepsze wykorzystanie swoich kwalifikacji — 48,2%
— przejście do środowiska o poprawniejszych stosunkach międzyludzkich — 9,5%
— otrzymanie wyższego stanowiska — 9,5%

Wymowne są opinie respondentów dotyczące stażu pracy w obsłudze technicznej, jaki jest niezbędny dla osiągnięcia biegłości w zawodzie. Według nich, wykwalifikowany specjalista technicznej obsługi komputerów powinien pracować w tym zawodzie:

— co najmniej rok — 0,9%
— co najmniej 2—3 lata — 52,3%
— co najmniej 3—4 lata — 45,0%

Zaledwie 1,8% respondentów uznało, że niekonieczne jest posiadanie takiego stażu.

Wynika stąd, że osiąganie wymaganej biegłości w zawodzie trwa stosunkowo długo; jest to związane z niewłaściwym doborem kadr, a także z brakiem w naszym kraju zorganizowanego systemu powszechnego kształcenia personelu technicznego. Ponadto potrzebne jest systematyczne samokształcenie. Należałoby więc stworzyć tej grupie pracowniczej szczególnie korzystne warunki, by jej przedstawiciele byli zainteresowani jak najdłuższym pozostaniem w zawodzie. Tymczasem znaczna część personelu obsługi technicznej nosi się z zamiarem odejścia do innej pracy.

Z drugiej strony — badanie potwierdziło hipotezę słabości i niedociągnięć w systemie obsługi technicznej ośrodków obliczeniowych.

Na pytanie, co ma znaczący wpływ na sprawność i jakość działania obsługi technicznej, odpowiedzi były następujące [2]):

— proces dystrybucji materiałów i części zamiennych — 70,9%
— ilościowy i jakościowy dobór personelu technicznego — 69,1%
— struktura wewnętrzna działu, wydziału — 24,5%
— wyposażenie w sprzęt diagnostyczny i kontrolno-pomiarowy — 17,7%

Przyczyny przestojów sprzętu informatycznego spowodowane czynnikami natury technicznej są — w opinii respondentów — następujące:

— brak części zamiennych, duża zawodność podzespołów i elementów elektronicznych i mechanicznych — 58,0%
— słabość lub brak serwisu fabrycznego — 44,1%
— niskie (zaniedbywane) kwalifikacje personelu technicznego — 40%
— niesystematyczna i nieodpowiednio prowadzona konserwacja i profilaktyka urządzeń — 28,2%
— niewłaściwa organizacja pracy personelu technicznego — 20,0%

Są to główne czynniki, które powodują małą efektywność i sprawność funkcjonowania obsługi technicznej. Można więc sądzić, że główne mankamenty tego stanu rzeczy są do przezwyciężenia w sferze organizacyjnej i odpowiednim doborze personelu technicznego. Poważnym problemem pozostaje nadal chroniczny brak części zamiennych i ich dystrybucja, lecz wymaga on odrębnego omówienia.

Dla prowadzenia usług w zakresie napraw sprzętu informatycznego stworzono też specjalistyczny serwis fabryczny, którego działalność została przez respondentów oceniona krytycznie. 61% badanych oceniło negatywnie jego działalność, uzasadniając swoją opinię następującymi mankamentami:

— brak szybkiej reakcji i mała operatywność na zgłaszane awarie sprzętu komputerowego
— zbyt długi okres realizacji zleceń
— niska jakość wykonywanych usług, nie zawsze należyte kwalifikacje personelu
— sprzedaż części — zamiast usług
— wysoki koszt usługi, nie zawsze uzasadniony
— utrzymywanie monoplu na dystrybucję części zamiennych.

Propozycje rozwiązań przedstawili sami respondenci, sugerując co następuje (w kolejności zgodnej z liczbą odpowiedzi):

1. Dokonać reorganizacji działów obsługi technicznej sprzętu informatycznego w ośrodkach obliczeniowych, podwyższając jakość procesu technologicznego konserwacji i profilaktyki.

2. Nadać właściwą rangę obsłudze technicznej, zawodowi inżyniera i technika — konserwatora sprzętu.

3. Powiązać odpowiedzialność za powierzony sprzęt z systemem bodźców materialnych.

4. Stworzyć regionalne (branżowe) magazyny części zamiennych, obsługiwane przez producenta, lub udostępnić magazyny poszczególnych delegatur serwisu fabrycznego bez koniecznych zleceń naprawczych, a tylko z zapłatą za części.

5. Stworzyć rzetelny, silny serwis obsługi sprzętu informatycznego prowadzony przez producentów sprzętu, którzy braliby na siebie odpowiedzialność za szybkość, terminowość i jakość świadczonych usług (a w przypadku niewywiązywania się z umów byliby karani).

Podstawową trudnością ośrodków jest brak regulacji prawnych w kwestiach zatrudnienia na stanowiskach technicznych pracowników o określonym profilu wykształcenia. Stąd też proces osiągania odpowiednich kwalifikacji i adaptacji zawodowej jest zbyt długi i kosztowny. Niska sprawność służb technicznych jest ponadto wynikiem niewłaściwej organizacji pracy w ośrodkach obliczeniowych, słabości ich struktury organizacyjnej i funkcjonalnej. A rozwiązanie tych akurat problemów nie wymaga nakładów finansowych czy zwiększania zatrudnienia bez usprawnień natury organizacyjnej.

JANUSZ MACIEJEWSKI
Warszawa

[2]) Respondenci mieli możliwość wybrania dwóch kategorii odpowiedzi spośród przedstawionych w ankiecie dlatego też suma procentów jest większa niż sto

Instytut Gospodarki Magazynowej

ul. Estkowskiego 6
61-755 Poznań
tel. 576-81

sprzeda

niekompletny komputer

MERA 400

bez jednostki centralnej,
modułu pamięci,
urządzeń wejścia—wyjścia

EO/392/K/85

Najnowsze tendencje w dziedzinie komputerowych systemów sterowania

Spośród niezliczonej ilości konferencji międzynarodowych, dotyczących zagadnień komputerowego sterowania, które odbyły się w 1983 roku, do tradycyjnego przeglądu dla INFORMATYKI [1]) wybrałem te, które wydały mi się nieco ważniejsze od innych, a próby uzyskania dostępu do materiałów zakończyły się choćby częściowym powodzeniem:

- **IFAC/IFIP Symposium on Real Time Digital Control Applications (RTDCA), Guadalajara (Meksyk). 15—21 stycznia**
- **12th IFAC/IFIP Workshop on Real Time Programming (RTP), Hatfield (Wielka Brytania), 29—31 marca**
- **Annual Meeting of the European Workshop on Industrial Computer Systems (EWICS), Graz (Austria), 6—8 kwietnia**
- **5th IFAC Workshop on Distributed Computer Control Systems (DCCS), Johannesburg (Płd. Afryka), 18—20 maja**
- **European Seminars on Local Area Networks (RTS/1 i RTS/2), Berlin Zachodni, 17—18 maja; Bruksela, 23—24 listopada.** Trudno też nie wspomnieć o Kongresie Międzynarodowej Federacji Przetwarzania Informacji:
- **9th World Computer Congress (IFIP), Paryż, 19—23 września.** Ze względu na podobieństwa tematyczne, warto też zwrócić uwagę na konferencję:
- **3rd Biennial Conference on Real-Time Computer Applications in Nuclear and Particle Physics, Berkeley (CA), USA, 16—20 maja,** której nie będę szczegółowo omawiał, gdyż jej prace wydrukowano w całości w stosunkowo łatwo dostępnym czasopiśmie [2]).

ARCHITEKTURA SYSTEMÓW

Wszelkie rozważania dotyczące komputerowych systemów sterowania muszą dotyczyć ich architektury. Choć do niedawna używano słów „konfiguracja sprzętu", a jeszcze wcześniej mówiono wręcz o samym sprzęcie, według obecnych poglądów wybór samego sprzętu czy technologii (przynajmniej w krajach wysoko rozwiniętych) jest sprawą drugorzędną; istotne jest natomiast właściwe zestawienie systemu komputerowego stosownie do przewidywanych funkcji. Pojawia się przy tym wiele dodatkowych zagadnień związanych z zestawieniem, połączeniem, rozłożeniem procesów i przydzieleniem im odpowiednich zadań.

Złożoność tych systemów jest tak znaczna, że mają one naturę sieci. Dość dawno stwierdzono, że przy łączeniu komputerów w większe jednostki niezbędny jest pewien stopień normalizacji — niezależnie od zastosowań i późniejszego wykorzystania tych zestawów (zmniejsza to koszty i ułatwia użytkowanie systemów). Od razu rozpoczęto też prace normalizacyjne w różnych organizacjach o zasięgu lokalnym lub międzynarodowym. Normalizacja okazała się jednak sprawą dość trudną i obecnie istnieje znaczna różnorodność rozwiązań sieciowych. Optymiści twierdzą jednak, że dokonanie podstawowych uzgodnień normalizacyjnych w tym zakresie jest już tylko kwestią czasu. Należy jednak podkreślić, że uzgodnienia międzynarodowe dotyczą wyłącznie tzw. systemów słabo związanych (ang. loosely coupled), do jakich zalicza się lokalne sieci komputerowe. Siecią lokalną nazywa się sieć, w której czas propagacji sygnału jest długi w porównaniu z czasem transmisji pojedynczego bitu i krótki w porównaniu z czasem transmisji typowej wiadomości [3]). Normalizacja systemów silnie związanych (ang. tightly coupled), wieloprocesorowych i wielomagistralowych, o których pisaliśmy w bieżącym roku [4]), jest jeszcze trudniejsza.

Większość obecnych rozwiązań opiera się w zasadzie na tzw. modelu referencyjnym ISO [5]), zwanym OSI (ang. Open System Interconnection), w którym wyróżniono siedem poziomów organizacji sieci (N. Zimmermann, D. Cohen, J. Postel, IFIP). Wprawdzie sam model nie jest całkowicie uniwersalny, niemniej stanowi dobrą ilustrację i punkt odniesienia dla istniejących i nowo opracowywanych sieci lokalnych. Normalizacja w tym zakresie jest jednak procesem dość długotrwałym. Od czasu poprzednich przeglądów niewiele się zmieniło w tej dziedzinie; choć autorzy odpowiednich referatów twierdzą, że normy są już prawie gotowe, w praktyce wielu uzgodnień jeszcze brakuje.

Najbardziej znane (H. Klessmann, RTS/1; M. Lobbele, RTS/2) i intensywnie studiowane w Polsce są projekty norm dotyczących sieci PROWAY (G. G. Wood, RTS/1) i IEEE 802 (T. Harrison, DCCS). Ponieważ podstawowe parametry tych sieci podawałem w poprzednich przeglądach, nie będę ich powtarzał. Warto jedynie dodać, że istnieje tendencja, aby sieć PROWAY — wzbudzająca także zainteresowanie w kraju [6]) — była w pełni zgodna z jedną z norm IEEE 802, których zbiór obejmuje kilka dokumentów. Chodzi tu o tzw. technikę token bus, którą nazwałbym dostępem znamieniowym, gdyż kontrola magistrali jest przydzielana stacjom przy użyciu specjalnego znamienia (ang. token). Techmika CSMA/CD (ang. carrier-sense multiple access with collision detection) — dostęp wspólny z wykrywaniem kolizji — jest już znana jako sieć ETHERNET (P. Kummer, RTS/1). Natomiast technika token ring jest normalizowana głównie ze względu na intensywne prace firmy IBM w kierunku opracowania sieci lokalnej opartej na tej zasadzie (W. Bux i in., RTS/1, EWICS). Wydaje się, że większość norm IEEE zostanie prawdopodobnie poddana procedurom normalizacyjnym ISO — do zatwierdzenia jako normy międzynarodowe.

Warto przyjrzeć się bliżej brytyjskiej normie CR82, opartej na sieci Cambridge Ring, opracowanej na Uniwersytecie w Cambridge [7]). Zasada działania tej sieci lokalnej, tzn. sposób przydziału dostępu do ośrodka transmisji, różni się od wymienionych powyżej i polega na przesyłaniu zmiennej liczby (od 1 do 15) tzw. szczelin (ang. slot), tj. pustych minipakietów o stałej długości (40 bitów), które mogą być przyjęte przez węzły (ang. node) w celu wysłania lub odbioru informacji. Węzeł składa się ze stacji, realizującej protokoły, i regeneratora (ang. repeater) sygnałów o częstotliwości 100 MHz (maksymalna odległość sąsiednich regeneratorów — 100 m). Jedna ze stacji, zwana monitorem, jest wyróżniona i służy do początkowego ustalenia struktury szczeliny, nadzorowania pracy sieci, usuwania straconych pakietów i gromadzenia informacji statystycznej o pracy sieci.

---

[1]) Poprzednie artykuły z tego cyklu opublikowano w numerach: 9/1980, 11—12/1981, 2/1983 i 12/1983

[2]) IEEE Transactions on Nuclear Science, Vol. 30, No. 5, October 1983

[3]) W. E. Burr: An Overview of the Proposed American National Standard for Local Distributed Data Interfaces. Communications of the ACM, Vol. 26, No. 8, pp. 554—561 (1983)

[4]) K. Rzymkowski: Mikrokomputerowy system modularny VME; R. Trechciński: MULTIBUS-II — wieloprocesorowy system modularny. INFORMATYKA, nr 6, 1984

[5]) ISO — International Organization for Standardization

[6]) A. Syryczyński: Zdecentralizowany mikroprocesorowy system automatyki kompleksowej MIR-PROWAY. Biuletyn MERA, nr 5, 1983

[7]) R. M. Needham, A. J. Herbert: The Cambridge Ring Distributed Computer System. Addison-Wesley, London, 1982

Na poziomie ramki nie ma ścisłej kontroli parzystości ani — powszechnie stosowanej — tzw. kontroli CRC [8]. Kopia wysłanego minipakietu jest zawsze przechowywana przez stację nadawczą i porównywana z powracającym minipakietem, zawierającym informację zwrotną od stacji odbiorczej. Protokoły kolejnych warstw sieci nie są dostosowane do modelu ISO, więc nie będę ich omawiał [9]. Sieci lokalne tego rodzaju istnieją już w kilkunastu uczelniach Wielkiej Brytanii, a przemysłowa produkcja urządzeń umożliwia stosowanie ich w automatyzacji badań i w przemyśle (K. S. Heard, D. Wanless, RTS/1).

Uważa się, że zasada pracy sieci Cambridge Ring jest znacznie prostsza niż sieci opartych na procedurze CSMA/CD, np. ETHERNET. Jednakże ta metoda dostępu jest stosowana poza Wielką Brytanią, dlatego proponuje się uwzględnienie jej także w normie IEEE 802 (P. Bryant, RTS/2).

Ogólnie, w dziedzinie normalizacji protokołów dla sieci lokalnych zostało do opracowania o wiele więcej norm niż przyjęto dotychczas. Głównie chodzi o wyższe warstwy modelu ISO. Najżywszą działalność w tej mierze prowadzi organizacja European Computer Manufacturers Association, która opublikowała już wiele zaleceń [10].

Obecnie panuje opinia, że najkorzystniejsze jest stosowanie sieci lokalnej ETHERNET (R. Metcalfe, IFIP), ponieważ została ona najszybciej znormalizowana (IEEE 802.3) i najwcześniej rozpoczęto produkcję układów sprzęgających w postaci kostek VLSI [11]. Należy jednak zaznaczyć, że struktura sieci ETHERNET nie w pełni odpowiada wymaganiom komputerowych systemów sterowania, szczególnie przy dużym obciążeniu (W. Erven i in., RTS/1). Nie ma wątpliwości, że kamieniem milowym na drodze do upowszechnienia lokalnych sieci komputerowych jest rozpoczęcie produkcji tanich sterowników w postaci układów scalonych realizujących odpowiednie protokoły. Nie musi to być jednak prawdziwe w odniesieniu do warunków krajowych. Nawet firma INTEL przed opracowaniem mikroukładów oferowała do sprzedaży dwupłytkowy sterownik do sieci ETHERNET [12].

Rozważania dotyczące architektury systemów rozłożonych przestrzennie trudno, oczywiście, sprowadzić do zagadnień normalizacji struktur sieciowych, choć te są dominujące. Oprócz takiego podejścia do komputerowych systemów sterowania, które wywodzi się z problematyki samych sieci komputerowych, istnieje inne, oparte na doświadczeniach praktycznych zdobytych przy sterowaniu procesami, formułowane na niższym poziomie abstrakcji i uogólniane w kategoriach matematycznych. Przykładowo — zastosowanie hierarchicznej dekompozycji złożonych systemów sterowania prowadzi do struktury wielowarstwowej, zbliżonej do modelu referencyjnego ISO (W. Gertenbach DCCS). Wyróżnia się w niej poziomy: sygnałów, bitów, ramki, pakietu, komunikatu, decyzji i koncepcji systemu. Odejście od tradycyjnych założeń dotyczących architektury komputerowych systemów sterowania doprowadziło innego autora (Th. Lalive d'Epinay, DCCS) do struktury sterowanej przepływem danych, opartej na zasadzie rozgłaszania (ang. broadcast principle) z adresowaniem źródeł danych. Jest interesujące, że rozwiązanie tego rodzaju stosuje się już w przemysłowych systemach produkowanych seryjnie, np. PARTNERBUS firmy BROWN BOVERY AND CO. [13].

Niekiedy okazuje się, że obojętne jest, które z dwóch wymienionych podejść zastosujemy do komputerowego systemu sterowania rozłożonego przestrzennie, gdyż różnice zanikają przy programowaniu (oczywiście — właściwym programowaniu). Takiego zdania są inni autorzy, zaczynający rozwiązywanie problemu od utworzenia systemu komunikacyjnego dla dowolnej konfiguracji (M. Sloman, J. Kramer, DCCS), co łączy się bezpośrednio z programowaniem.

PROGRAMOWANIE

Programowanie na poziomie określenia konfiguracji systemu stanowi właściwie specyfikację wymagań lub specyfikację oprogramowania. Zagadnienia specyfikacji systemów komputerowych są badane od niedawna, lecz mają już bogatą literaturę [14] (C. B. Jones, S. W. Smoliar, IFIP).

Specyfikacja konfiguracji oprogramowania w rozłożonym przestrzennie systemie czasu rzeczywistego obejmuje następujące elementy (J. Magee, J. Kramer, RTP):

— opis struktury logicznej, tj. składowych oprogramowania i ich powiązań logicznych
— opis struktury fizycznej, tj. składowych sprzętu i ich wzajemnych połączeń
— opis odwzorowania (ang. logical to physical mapping), tj. fizyczne położenie składowych logicznych.

Specyfikacja konfiguracji wyrażona w specjalnym języku formalnym umożliwia wprowadzanie zmian (ang. configuration change) przy użyciu edytora i ponowne zestawienie systemu za pomocą specjalnego programu (ang. rebuilder), a więc zapewnia dynamiczne zmiany konfiguracji, tzn. w czasie pracy systemu. Oczywiście, do tego celu, oprócz dodatkowych narzędzi, takich jak specjalny program zwany zarządcą konfiguracji (ang. configuration manager), potrzebne są pewne szczególne właściwości języka programowania (przede wszystkim możliwość rozłącznej kompilacji) i nowe funkcje systemu operacyjnego. W opisanym systemie odwzorowanie modułów programowych na stacje fizyczne (alokacja zadań) dokonuje się bezpośrednio przez przyporządkowanie w specyfikacji adresów stacji konkretnym modułom (ang. instance). Według autorów istotne jest, że opis połączeń i sposób konkretyzacji modułów nie powinien być wyrażony przez programistę (w języku programowania).

W systemach czasu rzeczywistego dominuje na ogół problematyka specyfikacji ukierunkowanych aplikacyjnie (ang. application oriented specifications) [15]. W cyklu istnienia systemu wyróżnia się specyfikację wymagań, specyfikację funkcjonalną i specyfikację projektu. Na ogół specyfikacje różnią się poziomem abstrakcji, stopniem formalizacji i liczbą uwzględnionych cech systemu. Pierwszym krokiem w kierunku specyfikacji wymagań jest dekompozycja systemu na podsystemy i wyróżnienie ich powiązań wzajemnych (oraz sprzężeń ze środowiskiem). Takiemu podziałowi odpowiada struktura dokumentu specyfikacyjnego, który powinien zawierać opis: środowiska (procesu), granicy systemu sterowania (tj. sprzężeń), budowy jego podsystemów i sprzężeń wewnętrznych oraz — wymagań szczegółowych na podsystemy. Istotne jest również oddzielne określenie wymagań implementacyjnych. Specyfikacja wymagań jest integralną częścią produktu i podstawowym źródłem informacji o systemie.

Jeden z kilku powstałych w RFN systemów automatycznego wspomagania specyfikacji wymagań [16], nazwany SARS (ang. System for Application Requirements Specification), omówiono nieco dokładniej (W. Epple, G. Koch, RTP). Jest on oparty na określonej metodologii umożliwiającej specyfikowanie wymagań użytkowych według omówionej metody i zawiera zbiór narzędzi wspomagających tworzenie i modyfikowanie specyfikacji oraz — operowanie nią.

---

[8]) CRC — Cyclic Redundancy Check

[9]) R. P. A. Collinson: The Cambridge Ring and Unix. Software Practice and Experience, Vol. 12, pp. 583—594 (1982)

[10]) Local Area Networks Standards, ECMA-81, 82, 90, 91

[11]) B. Metcalfe: Controller Transceiver Board Drives ETHERNET into PC Domain. Mini-Micro Systems, pp. 179—190 (January 1983)

[12]) J. P. Harakal: Putting ETHERNET Onboard the MULTIBUS. Computer Design, Vol. 20, No. 9, pp. 183—186 (1981)

[13]) H. Zimmermann: Decentralized Instrumentation and Control System with Bus Transmission in Power Plant. (pp. 386—390). From Electronics to Microelectronics, W. A. Kaiser, W. E. Proobster, eds., North-Holland, Amsterdam (1980)

[14]) C. B. Jones (Guest Editor): Special Section on Specification and Verification. IEEE Transactions on Software Engineering, Vol. 10, No. 2 (1984)

[15]) V. H. Haase, G. R. Koch (Guest Editors): Application Oriented Specifications. Computer, Vol. 15, No. 3 (1982); G. Koch, ed.: A Guide to Application Oriented Requirements Specifications. EWICS TC11 Report (1983)

[16]) J. Ludewig: ESPRESSO — A System for Process Control Software Specification. IEEE Transactions on Software Engineering, Vol. 9. No. 4, pp. 427—436 (1983)

Poszczególne podejścia stosowane w zagadnieniach specyfikacji wymagań różnią się w zależności od podstawowych rodzajów obiektów będących przedmiotem abstrakcji. Interesujący, bo dość naturalny, jest opis oparty na zdarzeniach (ang. event) i reakcjach na te zdarzenia (J. P. Thomesse, RTDCA). Zdarzenia, ich wystąpienia (ang. occurences) i zadania służące do przetwarzania zdarzeń, a także ich atrybuty — tworzą cały aparat pojęciowy użyteczny do formalnego opisu (a więc — specyfikacji) zachowania procesu przemysłowego — zarówno zachowania normalnego, jak i niepożądanego.

Należy podkreślić, że specyfikacja mająca praktyczną wartość musi wyodrębniać pewne właściwości systemu, pomijając inne — o mniejszym znaczeniu. Dlatego istnieje wiele metod specyfikacji, z których każda bierze za podstawę inne istotne cechy opisywanego systemu. Charakterystyczną właściwością różnych metod specyfikacji jest także stopień ich formalizacji, co z kolei ma wpływ na ich przydatność do analizy. Zautomatyzowana analiza składniowa i semantyczna znacznie ułatwia zrozumienie, ocenę przydatności i wykorzystanie specyfikacji (E. Knuth, EWICS).

Specyfikacja wymagań (tj. względnie precyzyjny i formalny, lecz łatwo zrozumiały opis zachowania i pożądanych wyników) powinna służyć do zaprojektowania systemu komputerowego i do napisania kodu programu, na ogół w języku innym od języka specyfikacji.

Do programowania systemów czasu rzeczywistego używa się nadal starych i dobrze znanych języków, jak BASIC i FORTRAN, choć często poddanych znacznym modyfikacjom. Dostosowuje się je do nowych wymagań. Przykładowo — BASIC przeżywa kolejną młodość [17] (A Lewis, B. Khan, RTP). Obecnie proponowane rozszerzenia, częściowo objęte propozycją nowej normy ANSI [18], wynikają z konieczności dostosowania języka do programowania systemów czasu rzeczywistego, lecz nie uwzględniają zagadnień sprzęgania i komunikacji — gdyż te informacje powinien wykorzystywać system zarządzania bazą danych, a nie sam kompilator. Przedstawiony system programowania, zwany IRTB (ang. Industrial Real Time BASIC), umożliwia utworzenie opisu zastosowania za pomocą czterech ortogonalnych zbiorów informacji:

— struktury i funkcji bazy danych
— synchronizacji działań współbieżnych
— kodu poszczególnych czynności
— specyfikacji konfiguracji sprzętu.

Większość tych informacji zapisuje się w metajęzyku, a dopiero później koduje formalnie w języku IRTB. Ważną cechą umożliwiającą programowanie procesów współbieżnych jest mnogość instrukcji służących do komunikacji: IN/OUT, SEND/RECEIVE, PUT/GET, READ/WRITE.

Inną tendencją w rozwoju BASICA, oprócz prób dostosowania go do programowania systemów współbieżnych i wieloprocesorowych, jest stopniowe tworzenie środowiska programowego (B. Khan, EWICS). Wiadomo również, że to samo dzieje się obecnie z FORTRANEM, choć tego języka nie omawiano explicite na żadnej z wymienionych konferencji.

O ile starsze języki z trudem poddają się procesom unowocześniania, to nowe — mają wszystkie ważniejsze właściwości wszczepione od chwili ich powstania. Problem polega natomiast na właściwej implementacji tych cech. Przykładowo — według doniesień różnych autorów, znaczne trudności występują przy implementacji spotkań w ADZIE, najważniejszej konstrukcji tego języka w odniesieniu do programowania systemów czasu rzeczywistego (P. Garetti i in., RTP). Trudności te zwiększają się przy wykonywaniu programu na zestawie wieloprocesorowym, w którym wywoływane zadania mogą znajdować się w różnych procesorach. Sprawa komplikuje się jeszcze bardziej, gdy niektóre z procesorów mają dostęp do wspólnej pamięci. Tego rodzaju problemy stanowią jedynie przykład bardziej różnorodnych kłopotów, związanych z programowaniem w tym języku. Mimo to, nadal uważa się, że jest to język przyszłości — bardzo dużo inwestuje się w dalsze prace z nim związane, szczególnie jeśli rzecz w rozwoju tzw. środowiska programowego, a więc narzędzi programistycznych (J. Ichbiah, IFIP).

Jednak ADA nie jest wyjątkiem. Na mapie języków programowania pojawiło się — równolegle z ADĄ — kilka innych języków tej klasy, zaprojektowanych z myślą o podobnych celach i mających w efekcie podobne właściwości. Jeden z nich, PRAXIS, powstał z przeznaczeniem do jednego konkretnego zastosowania, mianowicie — sterowania instalacją fuzji termojądrowej w Livermore (stan Kalifornia, USA). Opisywany w poprzednim przeglądzie system sterowania składa się z ponad 50 komputerów (w tym czterech — VAX-11 780). Konieczność opracowania nowego języka programowania wynikła z tych samych przyczyn, dla których powstała ADA — z potrzeby wyeliminowania języków asemblera, zwiększenia niezawodności oprogramowania, zmniejszenia kosztów pielęgnacji, polepszenia przenośności programów itd. W rezultacie równolegle z ADĄ powstał język, dla którego szybko opracowano kompilatory (PDP-11, VAX-11 780). Punktem przełomowym w istnieniu języka było niepodważalne potwierdzenie jego przydatności w programowaniu systemów czasu rzeczywistego przez opracowanie i umieszczenie w pamięci ROM (gdzie zastąpił 2 tys. linii kodu asemblerowego) program sterowania silnikiem krokowym przy użyciu mikroprocesora (G. Suski i in., DCCS).

Na podstawie analizy statystycznej 300 tys. linii kodu napisanego w tym języku oraz ankiet przeprowadzonych wśród programistów stwierdzono, że najczęściej używane są deklaracje danych całkowitych i bitowych oraz tradycyjne instrukcje sterujące. Zaskakujące jest natomiast to, że około 19% wszystkich linii programów źródłowych zawierało napisy, co wynika ze znacznej liczby komunikatów diagnostycznych i programowego wspomagania operatora. Za szczególnie cenną uznano możliwość operowania długimi nazwami identyfikatorów (ograniczonymi do 32 znaków). Te dwa spostrzeżenia są o tyle ciekawe, że wskazują na cechy języka pozostające pozornie bez żadnego związku z programowaniem systemów czasu rzeczywistego.

Warto również zaznaczyć, że w programach poddanych analizie statystycznej wystąpiło ok. 0,5 mln odwołań do identyfikatorów i ok. 4 tys. różnych typów zdefiniowanych przez użytkowników, co potwierdza konieczność ścisłej kontroli. Właśnie dzięki tej właściwości języka, tj. ścisłej typizacji i kontroli typów w czasie kompilacji, raz skompilowany program prawie nie wymagał dalszych interwencji podczas wykonywania go. Bardzo ważne okazało się też praktyczne opracowanie kompilatora z generatorem kodu pośredniego na język asemblera, który stanowił znaczną pomoc przy uruchamianiu programów. Natomiast znaczne trudności notowano wskutek rozłącznej kompilacji. Efektem zmian w jednym module często była niemożność ponownego skompilowania programu w uprzednio ustalonym porządku.

Wymienione właściwości języka PRAXIS i skutki jego stosowania rzucają więc pewne światło na skuteczność użycia ADY w komputerowych systemach sterowania. Najważniejsze wydaje się stwierdzenie, że oprogramowanie napisane w takim języku może działać niezawodnie już przy pierwszym wykonaniu programu, co nie zdarza się nigdy przy programowaniu w języku asemblera.

Językiem tej samej klasy jest CHILL, przedstawiony w poprzednich numerach INFORMATYKI [19]. Choć zaprojektowany i przeznaczony głównie do programowania systemów telekomunikacyjnych, może być stosowany z powodzeniem w różnych innych systemach sterowania, np. do sterowania sygnalizacją na kolei (N. Theuretzbacher, EWICS). Jednakże do tego celu wzbogacono go o takie np. właściwości, jak obsługa przeterminowania (ang. timeout handling), specjalne operacje wejścia-wyjścia, dostęp do bazy danych; choć bez ingerencji w składnię, lecz przy użyciu elementarnych funkcji (ang. primitive) systemu

[17]) W. Iszkowski, M. Maniecki: Standaryzacja języka BASIC, INFORMATYKA, nr 5, 6, 7—8 (1983); J. Bull, A. Lewis: Real-Time BASIC. SOFTWARE Practice and Experience, Vol. 13, No. 1075—1092 (1983)

[18]) American National Standard for BASIC, Draft Proposal, ANSI X3J2/84-26, June 1984

[19]) W. Udrycki, W. Wilczyński: CHILL — język programowania systemów komutacyjnych, INFORMATYKA, nr 9, 10, 11 (1984)

operacyjnego. W ten sposób usprawniono komunikację między zadaniami i obsługą wyjątków. Te funkcje systemu operacyjnego stanowią element całego środowiska programowego CHILLA.

Kompilator napisany w PASCALU (250 tys. linii) jest wielodocelowy, tzn. generuje kod dla różnych komputerów. W takiej sytuacji wszystkie dodatkowe funkcje systemu operacyjnego powinny dokładnie spełniać wymagania nałożone na sprzężenie z kompilatorem. Oprócz systemu wykonawczego (ang. run-time system), środowisko programowe tej implementacji CHILLA zawiera także wiele narzędzi służących do testowania programów zarówno w fazie statycznej, tzn. oddzielnych modułów w komputerze macierzystym, jak i dynamicznie, a więc — zintegrowanego programu na docelowym sprzęcie bez zakłócania działania całego systemu.

Zagadnieniom testowania oprogramowania poświęcono zresztą znacznie więcej uwagi na wszystkich konferencjach. Tworzy się nawet specjalne procesory do nadzoru poprawności wykonywania programów [20] (G. Schrott, T. Tempelmeier, RTP) i opracowuje różne metody testowania (S. Bologna, G. Dahll, U. Voges, J. Taylor, W. Ehrenberger i in., EWICS), lecz nie ma tu miejsca na ich szczegółowe omówienie.

Warto jednak zwrócić uwagę na diametralnie nieraz różniące się podejścia. Przykładowo — znana konstrukcja obsługi wyjątków, stosowana we wszystkich nowoczesnych językach programowania (także w omówionych powyżej), oprócz gorących zwolenników (R. W. Newton, EWICS); znalazła także przeciwników (G. Bull, R. Mitchell, RTP). Argumentują oni, że reguła obsługi wyjątków pokrywa się z logiką instrukcji GOTO, a więc przeczy zasadom programowania strukturalnego. W zamian proponuje się stosowanie instrukcji warunkowych w postaci bloków, twierdząc, że dzięki temu łatwiejsze jest testowanie, uruchamianie i pielęgnowanie programów.

BAZY DANYCH CZASU RZECZYWISTEGO

Wzrost złożoności komputerowych systemów sterowania i decentralizacja ich funkcji powodują znaczne zwiększenie i rozproszenie masywu danych, którymi operuje komputer. Właściwa organizacja tych danych, szczególnie w systemach czasu rzeczywistego, gdy w grę wchodzą nieprzewidywalne nieraz uwarunkowania czasowe, staje się problemem samym w sobie, do rozwiązania którego wymagane jest opracowanie zupełnie nowych metod [21]. Niemal na każdej konferencji przedstawiano referaty dotyczące tej problematyki.

Uważa się przy tym, że właściwości typowych systemów zarządzania bazą danych, zapewniających wydzielenie danych z programów, wspólne użytkowanie danych przez programy, nienaruszalność danych itp., nie wystarczają w systemach czasu rzeczywistego. Wymienia się szczególne cechy, istotne w systemach sterowania (K. Landwehr, H. Windauer, EWICS), np.:

— jednoczesny dostęp z programu z terminala
— wewnętrzna synchronizacja przy złożonym dostępie w jednej operacji
— ściśle ograniczony czas dostępu
— uwzględnianie priorytetu w czasie dostępu
— uwzględnianie nieprzewidywanych funkcji
— rozprzestrzenienie danych.

Proponuje się też niezwłoczne podjęcie prac nad standaryzacją systemów zarządzania bazami danych do sterowania procesami w kierunku opracowania modelu funkcyjnego, a następnie — języka operowania danymi, zapewniającego odpowiednie sprzężenia z oprogramowaniem sterującym. Jednakże okazuje się, że opracowanie praktycznych, a jednocześnie realistycznych modeli takich systemów nie jest sprawą prostą (G. Rodriguez, RTDCA).

Wraz z postępującą decentralizacją komputerowych systemów sterowania, skuteczne i zgodne z potrzebami zarządzania bazą danych w czasie rzeczywistym jest coraz trudniejsze. Do typowych problemów należy zbieranie danych z kilku tysięcy punktów pomiarowych rozłożonych wzdłuż linii o długości 1—10 km (E. R. Crichton, M. G. Willey, RTP), co oznacza konieczność jednoczesnego operowania około kilkuset tysiącami słów. Jeżeli nałożymy na to ograniczenia czasowe, to otrzymamy ilustrację skali trudności, które napotyka się w programowaniu i działaniu takich systemów sterowania. Obecnie próbuje się wypracować podejścia, które umożliwiałyby efektywne zarządzanie taką bazą danych. Jednym z zastosowanych rozwiązań jest koncepcja organizacji bazy danych w oparciu o tzw. człony (ang. item), stanowiące zbiory nierozdzielnych wartości danych. Każdy człon dzieli się na segmenty zróżnicowane według częstotliwości dostępu, np.:

— dane pierwszoplanowe, jak stan urządzeń, wartości dynamiczne zmienne (kilka słów)
— dane drugorzędne, jak ograniczenia, wartości zakresów, nazwy itp.
— dane trzeciorzędne, jak algorytmy (kilkaset słów).

Dzięki temu do każdego członu można zastosować inną strategię dostępu, nie naruszając integralności bazy danych. Innym kluczowym rozwiązaniem w tym systemie jest tworzenie wielokrotnych kopii tej samej bazy danych i przesyłanie ich potrzebującym podsystemom, pod warunkiem, że czyni to tylko właściciel (ang. owner) lub inny uprzywilejowany podsystem. Bardzo skuteczna w systemie zarządzania rozproszonymi bazami danych okazała się technika rozgłaszania (ang. broadcasting), tzn. przesyłania wiadomości przeznaczonej do odbioru przez wszystkie stacje, ponieważ znacznie zwiększa szybkość operowania danymi.

Poważnym problemem w systemach czasu rzeczywistego jest zapewnienie tzw. niesprzeczności, czyli — jednolitości danych (ang. consistency). Dane pomiarowe są zazwyczaj rozprzestrzenione na dużym obszarze i ulegają ciągłym zmianom, a dostęp do nich z różnych zadań wykonywanych współbieżnie jest asynchroniczny. Dlatego dla różnych zadań użytkowych te same dane mogą znaczyć co innego, gdyż mogą im odpowiadać inne wartości. Ważne są nie poszczególne dane, lecz złożony z nich obraz procesu jako całości. Przykładowo — jeżeli stan kilku przełączników jest próbkowany przez kilka zadań użytkowych, to wskutek różnic czasowych może się zdarzyć, że obraz uzyskany przez zadanie będzie kombinacją stanów, które nie mogą istnieć fizycznie. Z tych względów niezwykle istotne jest opracowanie metod zapewniających niesprzeczność i wiarygodność tych danych. Jedna ze stosowanych metod polega na „znakowaniu" wartości danych pomiarowych rzeczywistym czasem obserwacji. Definiując czasowy przedział ważności takich danych, można każdorazowo stwierdzić, czy aktualnie wykorzystywane informacje tworzą niesprzeczny obraz procesu (I. M. MacLeod, DCCS). Przy takim algorytmie zapewnienia jednolitości danych, istotnego znaczenia nabiera dostępność wszystkich podsystemów do rzeczywistych wartości czasu bezwzględnego.

Jednym z możliwych, choć nietypowych, obszarów zastosowań systemów baz danych tego rodzaju jest drogownictwo, a dokładniej — analiza stanu nawierzchni dróg na określonym terenie, w idealnym przypadku — na terenie całego kraju (S. Savoysky, RTP). Celem takiej analizy może być opracowanie doskonalszych materiałów, lepszych konstrukcji, a także określenie wzajemnych oddziaływań dróg ze środowiskiem naturalnym itd. Choć zastosowanie jest dość specyficzne, uwidocznia wszystkie istotne elementy systemu zarządzania bazą danych w czasie rzeczywistym: rozłożenie przestrzenne danych, potrzebę standaryzacji ważniejszych struktur danych, konieczność zapewnienia niesprzeczności danych itd. Dodatkowym utrudnieniem jest tu także rozłożenie przestrzenne obsługi na całym terytorium objętym systemem. Nie ulega więc wątpliwości, że do efektywnego zarządzenia tą bazą danych konieczne jest wypracowanie zupełnie nowych metod. Na razie autorom tego systemu udało się znaleźć adekwatny aparat jego opisu, lecz — jak twierdzą — najtrudniejsze zadanie czeka ich w fazie integrowania poszczególnych składników całego oprogramowania.

Problematyka baz danych, w różnych aspektach, była jednym z dominujących tematów ubiegłorocznego Kongresu IFIP (R. Bayer, P. Schnell, J. D. Ullman i in.).

[20]) G. Schrott, T. Tempelmeier: Messungen in einem Prozessrechnersystem mit einem eigenen Messrechner. Report TUM-I8202, Technische Universität München (Januar 1982)

[21]) T. Tempelmeier: Datenbanksysteme in der Prozessdatenverarbeitung. Regelungstechnische Praxis, Bd. 25, Nr 10, s. 432—440 (1983)

ZASTOSOWANIA

Spośród bardzo wielu prezentowanych zastosowań komputerowych systemów sterowania, w krótkim przeglądzie można omówić zaledwie kilka. Tradycyjnie największym uznaniem cieszą się komputery w przemyśle chemicznym, gdzie są stosowane m.in. do sterowania przebiegiem reakcji chemicznych (C. B. Brosilow), destylacji (D. E. Seborg, D. A. Mellichamp i in.) czy neutralizacji ścieków (A. B. Corripio), a także — w wielu innych przemysłach, jak hutniczy, celulozowo-papierniczy, cementowy, szklarski oraz — w energetyce (RTDCA). Wiele uwagi poświęca się zastosowaniu robotów do automatyzacji operacji dyskretnych w przemyśle wytwórczym (U. Rembold, EWICS; K. W. Plessman, DCCS; R. P. Paul, RTDCA).

Z racji zainteresowań zawodowych, najłatwiej jest mi omówić zastosowania komputerów w instalacjach jądrowych. Jak wspomniano w poprzednim przeglądzie, jest to dziedzina, w której — ze względu na skalę przedsięwzięć — szczególnie ostro uwidoczniają się braki dotychczas stosowanych rozwiązań i krystalizują się nowe idee dotyczące komputerowych systemów sterowania. W ubiegłym roku, nawet na Kongresie IFIP poświęcono tej tematyce oddzielny referat (P. Zanella), a w USA odbyła się trzecia z rzędu konferencja na ten temat [22]).

Rola komputerów w fizyce wielkich energii polega głównie na obsłudze eksperymentów, których celem jest badanie właściwości cząstek elementarnych przyspieszanych w wielkich akceleratorach o średnicy dochodzącej do kilku kilometrów. Pewne wyobrażenie o skali tych eksperymentów może dać informacja, że pojedynczy eksperyment, a właściwie ciąg doświadczeń, jest przygotowywany przez kilka lat, a wykonywany przez następnych kilka, podczas których generowanych jest ok. 10^7 zdarzeń, tj. informacji o każdorazowej objętości około kilkudziesięciu kilobajtów. W ciągu sekundy, podczas której następuje interakcja cząstek, może zajść około stu zdarzeń wartych odnotowania. W najbardziej skomplikowanym i interesującym eksperymencie, tzw. UA1 (przeprowadzanym w laboratoriach CERN, pod Genewą), potwierdzającym przewidziane teoretycznie istnienie cząstek elementarnych, tzw. bozonów, bierze udział ok. 120 naukowców z jedenastu laboratoriów różnych krajów. Po kolizji cząstek, protonów i antyprotonów pędzących naprzeciw siebie z ogromną energią, jednorazowa porcja danych z detektorów promieniowania wynosi 1,6 MB. Ażeby zredukować tę informację do ilości możliwej do przetworzenia w wymaganym czasie, w instalacji komputerowej wokół obiektu działa ponad 300 mikroprocesorów (M6800 i 8X300). Na wyższych poziomach pracują mikrokomputery 16-bitowe (jak MOTOROLA 68000), mini- (PDP-11 i HP21MX) i superminikomputery (VAX-11/780, NORD 500), których rola polega na przetwarzaniu, analizowaniu i obrazowaniu informacji w trybie bezpośrednim. Tak wyselekcjonowane zdarzenia, które mogą mieć znaczenie dla fizyków, są poddawane dalszej analizie na wielkich komputerach (CDC 7600, IBM 3081 i in.) sprzężonych w sieć. Spośród kilku sieci komputerowych opracowanych w CERN (OMNET, CERNET, INDEX) obecnie najbardziej fascynująca jest realizacja sieci satelitarnej zwanej STELLA [23]) (N. Celandroni i in., RTS/1). Sens stosowania transmisji satelitarnej wynika stąd, że w eksperymentach bierze udział kilka tysięcy fizyków z około stu laboratoriów Zachodniej Europy. Ogromne ilości danych (w latach 1978—1980 zarejestrowano je na 10 tys. taśm magnetycznych) generowanych przez detektory promieniowania są analizowane przez eksperymentatorów w trybie rozłącznym (ang. off-line). Przesyłanie automatyczne tych danych i wyników obliczeń prowadzi do znacznych oszczędności. Oprócz laboratoriów fizycznych, w przedsięwzięciu bierze udział Europejska Agencja Kosmiczna (European Space Agency).

Ażeby tylko pobieżnie naszkicować ogrom przyjętych nowoczesnych rozwiązań, które mogą sprostać wymaganiom, nie starczyłoby całego numeru INFORMATYKI. Uważam jednak, że należy choćby wymienić niektóre z nich, aby zaznajomić Czytelników z rzeczywiście najnowszymi tendencjami w tej dziedzinie. Pełny przegląd tej tematyki przedstawia się w dwuletnim cyklu na konferencjach organizowanych w USA [2,22]). Po lekturze tych artykułów można zrozumieć, dlaczego system CAMAC — choć technicznie nie najdoskonalszy — uzyskał tak ogromną popularność. Przy kilkuset mikrokomputerach w zestawie i wielokrotnie większej liczbie bloków funkcjonalnych, których używają dziesiątki osób jednocześnie, normalizacja jest koniecznością. Obecnie jednak do szerokiego użytku w laboratoriach jądrowych Ameryki Północnej, Europy Zachodniej i Japonii wchodzi nowy system modularny, dziesięciokrotnie szybszy od CAMACA i wieloprocesorowy, tzw. FASTBUS. Jednakże jest to system o bardzo dużym stopniu złożoności i opracowanie samych norm mechanicznych, elektrycznych i logicznych, co trwało kilka lat, nie wystarcza. Konieczne są standardy użytkowe, a więc normy dotyczące testowania modułów, sprzęgania systemu z komputerami i programowania.

* * *

Zdaję sobie sprawę, że przedstawiony opis jest bardzo powierzchowny. Pełen obraz tendencji w dziedzinie komputerowych systemów sterowania można uzyskać jedynie po lekturze kilkunastu tomów materiałów konferencyjnych. Sądzę jednak, że selekcja materiału była na tyle trafna, iż najważniejsze zagadnienia nie zostały pominięte, a bardziej dociekliwy Czytelnik sięgnie po interesujące Go informacje do oryginalnych publikacji.

JANUSZ ZALEWSKI

[22]) D. W. O'Brien: Review of the 3rd Biennial Conference on Real-Time Computer Applications in Nuclear and Particle Physics. IEEE Transactions on Uuclear Science, Vol. 31, No. 1, pp. 178—179 (1984)

[23]) N. Celandroni et al.: STELLA — Satellite Interconnection of Local Area Networks. (pp. 425—430). Proc. 6th Conf. on Computer Communications, London, 7—10 September 1982

Zakład Przemysłu Metalowego H. CEGIELSKI

Odlewnia Żeliwa

63-100 Srem, ul. Staszica 1, tel. 741

zatrudni informatyków i elektroników

Zdolnym pracownikom gwarantujemy — po umownym okresie próbnym — uzyskanie mieszkania rodzinnego. Zakład dysponuje hotelem pracowniczym I kategorii, stołówką, obiektami rekreacyjno-sportowymi, ośrodkiem żeglarskim, działkami pracowniczymi, działkami pod budowę domków jednorodzinnych oraz ośrodkami wypoczynkowymi.

Prosimy nadsyłać oferty wraz z życiorysem oraz ksero-kopią dyplomu.

EO/390/K/85

CSK–Komputer Studio Kajkowscy

81-505 GDYNIA ORŁOWO ul. Balladyny 3B, tel. 29-00-18

Komputer osobisty może być przydatny niemal na każdym stanowisku pracy. Wymaga jednak odpowiedniego oprogramowania użytkowego. W ramach tego oprogramowania oferujemy zainteresowanym dostawę uniwersalnych pakietów programowych:

BANK DANYCH CSK, TABPLAN CSK, TEKST CSK, TRANSCOM CSK

To doskonałe narzędzia pracy dla każdego. Aby z nich korzystać, nie trzeba być informatykiem! Zupełnie samodzielnie można tworzyć złożone systemy zarządzania przedsiębiorstwem, k a ż d y m p r z e d s i ę b i o r s t w e m; nawet najbardziej specyficzne uwarunkowania nie są przeszkodą.
To jednak jeszcze nie wszystko... Kiedy dotychczasowe problemy łatwo i szybko zostały rozwiązane — pojawiają się zupełnie nowe. Można wtedy bez kłopotów samemu udoskonalić dotychczasowy system!

BANK DANYCH CSK, TABPLAN CSK, TEKST CSK, TRANSCOM CSK

składają się w zakładowe systemy płacowe, osobowe, finansowo-księgowe lub magazynowe. Korzystając z nich, z łatwością można prowadzić planowanie, kalkulacje i sprawozdawczość. Można też sporządzać kosztorysy i oferty, a nawet prowadzić „automatyczną" korespondencję czy redagować dowolne teksty. Można wreszcie skorzystać z już zgromadzonych zasobów na komputerze ODRA (pod nadzorem systemu GEORGE-3), wykorzystując komputer osobisty jako inteligentny terminal — stację lub emulator TTY.

Nowość:

Oferujemy system operacyjny kompatybilny z CP/M 2.2. dla mikrokomputerów ROBOTRON 5120/5130 oraz systemy finansowo-księgowe FK dla dowolnych mikrokomputerów

---

Szczegółowych informacji udziela:

CSK–Komputer Studio Kajkowscy

81-505 GDYNIA ORŁOWO, ul. Balladyny 3B, tel. 29-00-18

EO/129/k/85

Rodzaje drukarek

Technologie druku, konstrukcja i parametry drukarek zmieniają się obecnie bardzo szybko. Wzrasta jakość, precyzja i prędkość druku, rodzą się nowe możliwości odwzorowań graficznych. Artykuł zawiera zwięzły przegląd nowości realizowanych w drukarkach uchodzących za tradycyjne, omówione zostaną również nowe koncepcje rozwiązań.

Drukarki mozaikowe (ang. dotmatrix printers)

Drukarki te należą do klasy drukarek uderzeniowych (ang. impact printers). Powoduje to oczywiście ich większą hałaśliwość (65—85 db) w porównaniu z drukarkami bezuderzeniowymi, jakimi są np. drukarki laserowe czy termiczne (ok. 50 db). W drukarkach mozaikowych jakość druku zależy nie tylko od ich parametrów konstrukcyjnych lecz również od stanu zużycia taśmy barwiącej. Inną wadą drukarek mozaikowych jest „komputerowy" wygląd druku — jak nazywają to użytkownicy (widoczne kropki tworzące obraz znaku). Ulepszenia idą więc w kierunku zagęszczenia kropek.

Głowica drukarki mozaikowej zawiera cienkie igiełki, które są „wystrzeliwane" za pomocą obwodu elektrycznego w kierunku taśmy barwiącej. Taśma dociskana przez igiełki do papieru tworzy na nim znak. Impulsy pobudzające obwód elektryczny wysyłane są z generatora znaków zapisanego w pamięci stałej (ROM) drukarki. Ruch powrotny głowicy jest również ruchem roboczym, co zwiększa prędkość druku w porównaniu do drukarek, w których ruch powrotny głowicy jest ruchem jałowym. Drukarki mozaikowe są wyposażone ponadto w specjalne elementy logiczne wyszukujące najkrótszą drogę głowicy między kolejnymi punktami druku.

Jeszcze niedawno liczba igiełek w głowicy drukarki mozaikowej wynosiła od 5 × 7 do 7 × 7, obecnie produkowane są głowice o większym zagęszczeniu: 9 × 7 i 9 × 9 oraz 18 lub 24 na 7 lub 9 — dla druku o podwyższonej jakości. Zastosowanie, prawie nakładających się pionowych szeregów igiełek umożliwiło wytworzenie przesuniętych nieznacznie kropek, przez co druk stał się bardziej „ciągły". Podwójne szeregi, bardzo cienkich igiełek pozwoliły również drukować fragmenty liter wystające poza dolny zarys linii — takie jak np. ogonki w „g" czy „y".

W trybie pracy zwanym programowaną grafiką drukarki mozaikowe mogą wytworzyć około 80 × 80 kropek/cal^2, a nawet i więcej o ile użyje się specjalnego oprogramowania graficznego. Metoda druku za pomocą przesuniętych kropek polepsza jego jakość lecz znacznie zmniejsza prędkość druku. Dlatego też prawie wszystkie, oprócz najtańszych, drukarki mozaikowe mogą pracować w dwóch trybach. W pierwszym trybie osiąga się wysoką jakość druku kosztem prędkości (40—80 znaków/s), a w drugim — większą prędkość lecz pogorszenie czytelności druku (160—200 znaków/s).

Najtańsze drukarki mozaikowe, o głowicach 7 × 7 albo 9 × 7 kosztują obecnie od 200 do 500 dol. Jeszcze dwa lata temu drukarka taka kosztowała ok. 1200 dol.

Drukarki rozetkowe (ang. daisywheel printers)

Innym typem drukarek uderzeniowych są drukarki rozetkowe oraz tulejowe (ang. thimble printers). Drukarki te wytwarzają druk wysokiej jakości i działają podobnie. Zamiast obwodów elektrycznych i igiełek, jak w drukarkach mozaikowych, drukarki rozetkowe mają wirującą rozetkę z elastycznymi pręcikami i młoteczek. Na końcach pręcików wygrawerowane są znaki. Impuls sterujący powoduje ustawienie odpowiedniego pręcika rozetki nad taśmą barwiącą i uderzenie młoteczka. Drukarki tulejowe działają na tej samej zasadzie, a różnią się jedynie kształtem tarczy ze znakami.

Jeszcze niedawno drukarki rozetkowe były bardzo hałaśliwe i stosunkowo powolne, bowiem ich prędkość zamykała się w granicach 12—18 znaków/s. W ciągu ostatniego roku zaszło wiele zmian. Najpierw, na rynku, pojawiła się drukarka Smith-Corona TP-1 będąca pierwszą, tanią (800 dol.) i dosyć szybką drukarką rozetkową, przystosowaną do współpracy z komputerem osobistym. W ślad za tym coraz więcej producentów zaczęło oferować drukarki rozetkowe o niezłych parametrach, przeznaczone do współpracy z minikomputerami.

Istotne polepszenie jakości tych drukarek (oraz zmniejszenie ceny) nastąpiło w wyniku wprowadzenia mikroprocesorowego sterowania ruchem elementów drukujących oraz zastosowania, podobnie jak w drukarkach mozaikowych, metody wyszukiwania najkrótszej drogi głowicy między kolejnymi znakami. Te modyfikacje umożliwiły stosowanie drukarek rozetkowych również tam, gdzie dotąd dominował druk mozaikowy — a mianowicie w grafice wysokiej rozdzielczości (np. kreślenie map).

Drukarki rozetkowe są jednak nadal stosunkowo powolne (12—40 znaków/s), co stanowi od 1/2 do 1/4 prędkości najwolniejszych drukarek mozaikowych. Niepodważalna zaleta drukarek rozetkowych (i tulejowych) jest natomiast doskonała jakość druku. Ponadto, wymieniając rozetkę, można szybko i łatwo zmieniać repertuar znaków. Drukarki rozetkowe, tak jak wszystkie drukarki typu uderzeniowego są niestety dość głośne.

Drukarki strumieniowe (ang. ink-jet printers)

W drukarkach strumieniowych stosowane są dwie metody odwzorowania znaków, a mianowicie technika ciągłego strumienia (ang. continuous stream), oraz technika „kropli na żądanie" (ang. drop on demand). W technice ciągłego strumienia, strumień atramentu wyrzucany jest przez dyszę w ruchomej głowicy, a litery są formowane w efekcie ruchu tej głowicy. Drukarki działające w oparciu o technikę ciągłego strumienia są konstrukcyjnie bardzo złożone, a w związku z tym ich cena przekracza 30 000 dol. Metoda kroplowa jest znacznie mniej kosztowna w realizacji, a więc drukarki się znacznie tańsze, umożliwiając zastosowanie w komputerach osobistych.

Głowica drukarki strumieniowej o technice kroplowej, składa się z wielu dysz, których liczba i rozmieszczenie zależy od typu i klasy drukarki. Dysze połączone są z kanałami doprowadzającymi ze zbiornika atrament. Wewnątrz każdego kanału znajduje się rurka wykonana z piezoelektrycznego kryształu. Stymulacja, za pomocą impulsu elektrycznego powoduje rozszerzenie kryształu, a w konsekwencji wzrost ciśnienia w kanale. Zwiększone ciśnienie tłoczy atrament w kierunku dyszy, gdzie tworzy się mikroskopijna kropla. Gdy kryształ się kurczy — ciśnienie maleje, kropla odrywa się od dyszy i spada na powierzchnię papieru. Kropelki układają się w kształt odwzorowujący literę. W dyszach, które w danej chwili nie powinny generować kropel — podciśnienie hamuje wyrzucanie atramentu. Podobnie jak w drukarkach mozaikowych, ruch powrotny głowicy jest ruchem roboczym.

Główną zaletą drukarki strumieniowej, opartej zarówno na metodzie kroplowej jak i strumienia ciągłego, jest mała hałaśliwość (brak mechanicznego uderzenia w papier) oraz niski koszt eksploatacji (drukarka ma zaledwie kilka ruchomych części). Ponieważ głowica pisząca nie dotyka papieru — jej żywotność jest bardzo duża i wynosi ok. 10 mld znaków. Dla porównania, żywotność głowicy drukarki mozaikowej wynosi ok. 200 mln znaków. Drukarki strumieniowe są, poza tym bardzo szybkie, ich prędkość wynosi 150—300 znaków/s, podczas gdy prędkość drukarki mozaikowej wynosi obecnie ok. 200 znaków/s. Wadą drukarek strumieniowych, opartych na metodzie kroplowej, jest brak możliwości jednoczesnego wytworzenia kilku kropli oraz wyraźny zarys kropek tworzących znak. Ponadto, uzyskanie druku wysokiej jakości wymaga użycia specjalnego papieru absorpcyjnego, chociaż stosowanie zwykłego papieru daje zupełnie dobre efekty.

Należy podkreślić, że zasada działania drukarek strumieniowych stwarza duże możliwości graficzne. Precyzyjne pozycjonowanie głowicy pozwala wykonywać cieniowane rysunki zawierające bardzo dużo szczegółów, np. drukarka SIMENS PT-88 może być programowana w przyrostach rzędu 1/72 cala. Druk może być wielobarwny. Specjalny nabój zawierający cztery zbiorniki kolorowego atramentu umożliwia nie tylko druk w barwach podstawowych, lecz również tworzenie nowych barw w efekcie kombinacji

kolorów podstawowych. W drukarkach mozaikowych (i rozetkowych) druk barwny wymaga użycia specjalnych taśm. Wypisanie znaków w jednym kolorze wymaga tu jednego przejścia głowicy, a więc druk wiersza złożonego z różnokolorowych znaków wymaga wielu przejść głowicy, co znacznie zmniejsza prędkości pracy.

W drukarkach strumieniowych wszystkie kolory są nakładane jednocześnie. Mikroprocesor sterujący głowicą może wprawić ją w tak precyzyjne drgania, że wyrzucane krople atramentu mieszają się ze sobą w różnym stopniu tworząc ciemniejsze i jaśniejsze odcienie barw.

Jak wspomniano, drukarki strumieniowe są stosowane coraz częściej do komputerów osobistych. Przoduje tu amerykańska filia koncernu SIMENS, produkująca drukarki strumieniowe serii PT88-T2 oraz 2717-M203.

Jeszcze dwa lata temu drukarka strumieniowa średniej klasy kosztowała od 2500 do 6000 dol. Dziś koszt drukarki strumieniowej dobrej klasy, wynosi ok. 900 dol., co odpowiada cenie drukarki mozaikowej podobnej klasy.

Drukarki laserowe (ang. laser printers)

Druk wytwarzany przez te drukarki dorównuje jakością najlepszym technikom offsetowym. Przykładem drukarki laserowej może być XEROX 5700, która może zapamiętać i odtworzyć do 256 różnych znaków, z których każdy składa się od 6 do 24 kropek identyfilaryjnych. Techniką laserową można drukować po obu stronach zwykłego papieru. W ciągu jednej minuty drukarka taka jest w stanie wydrukować aż 43 strony. Laserowy system generowania obrazu ma rozdzielczość wynoszącą do 90 tys. kropek na 1 cal², przy czym położenie każdej kropki jest kontrolowane. Daje to oczywiście niezwykłą precyzję druku. Kopie wydruku mogą być tworzone metodą kserograficzną. Są to urządzenia nadal bardzo drogie, np. wspomniana drukarka XEROX 5700 w wersji podstawowej kosztuje ok. 65 tys. dol.

Drukarki termiczne (ang. thermal printers)

Głowice drukarek termicznych przypominają budową głowice drukarek mozaikowych. Nie używa się tu jednak taśm barwiących ani atramentu. Druk oparty jest na zasadzie przekazania ciepła z głowicy termicznej na papier powleczony specjalną substancją (papier termiczny). Głowica składa się z kilku (7—8) miniaturowych elementów grzejnych. Impulsy sterujące powodują rozgrzanie odpowiednich elementów. Ponieważ głowica jest stale dociśnięta do papieru, w miejscach styku ciepłych elementów z papierem zachodzi reakcja termochemiczna, w wyniku której papier zmienia barwę. Ruch głowicy odbywa się tylko w jedną stronę.

Drukarki termiczne należą do najtańszych, ale ich druk jest najgorszej jakości. Inną wadą była do niedawna wysoka cena papieru termicznego. Obecnie, najnowsze typy drukarek termicznych mogą drukować już na zwykłym papierze, rolę emulsji termochemicznej pełni bowiem specjalna taśma umieszczana między głowicą a papierem. Niektóre drukarki termiczne mogą wytwarzać już nawet druk barwny.

Drukarki termiczne serii MTP i STP firmy SEIKO INSTRUMENTS ELECTRONICS LTD mają prędkość druku 24—35 znaków/s. W drukarkach tych jest stosowany papier o szerokości od 38 do 216 mm, umożliwiający uzyskanie w jednym wierszu od 13 do 80 znaków alfanumerycznych w drukarkach znakowych oraz od 128 do 640 kropek w drukarkach typu graficznego. Producent określa średni czas bezawaryjnej pracy drukarek na minimum 0,5 mln wierszy.

Opracowała

TERESA WILCZEK

na podstawie „Radio-Electronics" kwiecień 1984 oraz materiałów firmowych

Ogłoszenia • Ogłoszenia

Spółdzielnia Rzemieślnicza Elektromechaników „Elmech", 00-312 Warszawa, ul. Dobra 56 oferuje owijarki mechaniczne (pistoletowe) oraz końcówki do owijarek (zachodnich) do drutu o średnicy 0,25—0,5 mm. Zamówienia prosimy kierować pod adresem Spółdzielni. Informacje telefoniczne pod nr. 22-94-46 lub 50-02-34. *EO/393/K/85*

Sprzedam mikrokomputer osobisty SHARP PC1500 z wyświetlaczem, 8K RAM, drukarka graficzna czterokolorowa CE-150. Idealny do zastosowań inżynieryjnych i ekonomicznych. Warszawa, tel. 49-64-46. *EO/394/K/85*

Ogłoszenia • Ogłoszenia

„NEW GENERATION COMPUTING"

W 1983 roku pojawił się pierwszy numer czasopisma stanowiącego forum wymiany informacji w dziedzinie systemów komputerowych Piątej Generacji (SKPG) — kwartalnika NEW GENERATION COMPUTING. Główne działy czasopisma to: podstawy teoretyczne, język programowania, architektura komputerów, technologia projektowania VLSI, przetwarzanie wiedzy.

Pierwszy numer rozpoczyna artykuł wstępny prof. T. Moto-Oki z Uniwersytetu Tokijskiego. „Informatyka — pisze T. Moto-Oka — natrafiła w swym rozwoju na bariery; konwencjonalne architektury maszyn cyfrowych, oparte na koncepcjach von Neumana i Turinga, napotykały liczne trudności. Najpoważniejsze to: niska produktywność oprogramowania, niewystarczająca efektywność przetwarzania danych nienumerycznych i słabe sprzęgi człowiek-maszyna. Obiecujące wyniki badań w takich dziedzinach, jak sztuczna inteligencja, inżynieria oprogramowania, architektura komputerów oraz bazy danych — wskazują na możliwość budowy komputerów nowej generacji, które pokonają te trudności. Postęp zapewnia opracowanie dojrzałych wersji języków LISP i PROLOG oraz rozwój nowych technologii programowania, takich jak abstrakcja danych i programowanie ukierunkowane na obiekty. Istotny wkład w rozwój komputerów nowej generacji wnoszą również ostatnie osiągnięcia w dziedzinie architektury komputerów, takie jak architektura przepływu danych i architektura redukcji danych oraz — dalszy postęp w dziedzinie technologii VLSI".

Dyrektor Centrum Badawczego ICOT w Japonii, dr K. Fuchi, wyjaśnia w kolejnym artykule nieporozumienia narosłe wokół projektu SKPG. Podkreśla on długofalowy charakter projektu oraz konieczność skoncentrowania się na badaniach podstawowych, z całkowitym pominięciem doraźnych celów handlowych. Główny nacisk położony zostanie na programowanie logiczne jako — zdaniem autora — brakujące ogniwo łączące różne dziedziny informatyki: sztuczną inteligencję, inżynierię wiedzy, inżynierię oprogramowania, nowe architektury komputerów i inne.

T. Chikayama, z Centrum Badawczego ICOT, omawia w następnym artykule pierwszy trzyletni etap realizacji projektu SKPG, w którym opracowane zostaną bardzo wydajne komputery osobiste, nazywane maszynami wnioskującymi. Maszyny te wyposażone są w język maszynowy wysokiego poziomu KLO (ang. Kernel Language O), oparty o rozszerzoną wersję PROLOGU. Do opracowania oprogramowania podstawowego wykorzystany zostanie język ESP. W pierwszym etapie opracowane zostaną systemy operacyjne i oprogramowanie użytkowe dla tych maszyn. W następnych etapach posłużą one jako narzędzia umożliwiające opracowanie bardziej zaawansowanych systemów.

Artykuł E. Shapiro i A. Takeuchi wskazuje, że podstawowe operacje języków programowania ukierunkowanych na obiekty (tworzenie obiektów, wysyłanie i odbiór wiadomości, modyfikacja stanu obiektu i tworzenie hierarchii klas) mogą być implementowane w sposób naturalny we współbieżnym PROLOGU.

Wreszcie H. Nakashima i N. Suzuki w artykule „Abstrakcja danych w PROLOGU/KR" opisują język programowania logicznego PROLOG/KR, umożliwiający opracowanie dużych systemów oprogramowania.

Poza głównymi artykułami, pierwszy numer czasopisma uzupełniają krótkie komunikaty omawiające konstrukcje nowych komputerów i systemów operacyjnych oraz interesujące rozszerzenia języka PROLOG.

Rozprowadzaniem czasopisma (roczna prenumerata — ok. 80 dol.) zajmuje się Springer Verlag (Heidelberg, RFN).

W.E.B.

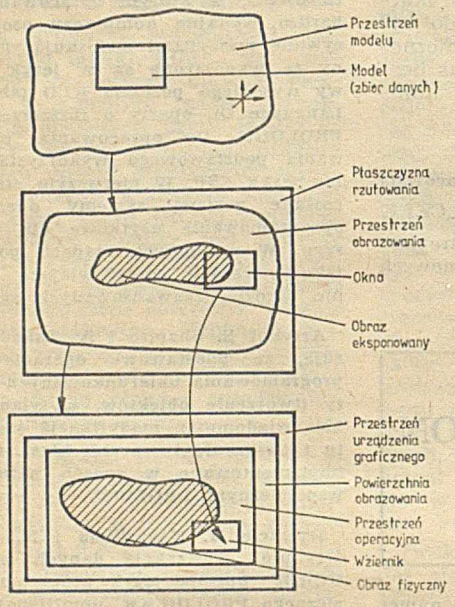
Słownik pojęć i terminów z dziedziny grafiki komputerowej

Śledząc coraz bogatszą literaturę z dziedziny grafiki komputerowej, obserwujemy narastającą niejednoznaczność oraz nieporządek pojęciowy i terminologiczny. Odnosi się to zarówno do publikacji w językach obcych (głównie angielskim), jak też w polskim. W coraz częściej pojawiających się opracowaniach w języku polskim [1]), w tym również w jednym podręczniku nt. grafiki komputerowej [2]) wyraźnie brakuje jednolitego spojrzenia na tę dziedzinę wiedzy. Nie ma w niej ani jednolitego słownictwa, ani spójności pojęciowej. Rezultatem tego krytycznego spojrzenia była myśl przygotowania niniejszego słownika [3]).

Celem naszej pracy było więc zebranie i zaproponowanie najpowszechniej używanych, najtrafniejszych i najlepiej oddających istotę rzeczy polskich nazw, terminów i pojęć oraz — ich wzajemnych związków.

Opisując terminy i pojęcia występujące w tej dziedzinie korzystaliśmy z wielu publikacji, przede wszystkim w języku angielskim, a zwłaszcza z propozycji słownika graficznego [4]), opublikowanego przez SHARE Inc., oraz z projektu normy ISO, dotyczącej grafiki komputerowej [5]). Ostateczny jednak kształt proponowanego słownika wynikł z doświadczeń nagromadzonych w ciągu kilkunastoletniego okresu badań w dziedzinie grafiki komputerowej.

Przyjęta na potrzeby słownika koncepcja grafiki komputerowej (i naszym zdaniem — zgodna z rzeczywistością) opiera się na wzajemnych relacjach odwzorowania i zawierania, uwidocznionych na rysunku.



W nawiasach podajemy angielskie odpowiedniki przyjętych polskich terminów. Wielkimi literami pisane są określenia przedstawione w „Słowniku".

ADRESOWALNOŚĆ (addressability) — liczba PUNKTÓW ADRESOWALNYCH PRZESTRZENI URZĄDZENIA GRAFICZNEGO; obszar adresowy (p. MACIERZ ADRESOWALNA).

ATRYBUT (attribute) — charakterystyczna cecha ELEMENTU OBRAZU, np. kolor, POZIOM INTENSYWNOŚCI ŚWIECENIA, WYKRYWALNOŚĆ.

BUFOR OBRAZU, PAMIĘĆ OBRAZU (display buffer) — obszar pamięci, który przechowuje ROZKAZY GRAFICZNE i dane, potrzebne do generowania OBRAZU FIZYCZNEGO. Bufor może być częścią pamięci operacyjnej komputera lub urządzeń specjalnych.

CYKL ODNAWIANIA, CYKL ODŚWIEŻANIA (display cycle) — czas niezbędny do wygenerowania i wykonania ciągu przesunięć strumienia elektronów potrzebnych do jednokrotnego odtworzenia OBRAZU FIZYCZNEGO na ekranie LAMPY Z ODNAWIANIEM.

CZAS WYMAZYWANIA (erase time) — czas niezbędny do usunięcia obrazu z ekranu, odnosi się do monitorów graficznych z LAMPĄ PAMIĘCIOWĄ).

CZĘSTOTLIWOŚĆ ODNAWIANIA, CZĘSTOTLIWOŚĆ ODŚWIEŻANIA (refresh rate) — liczba cykli na sekundę, z jaką należy ODNAWIAĆ obraz na monitorze, aby uzyskać wrażenie jego stabilności (odwrotność cyklu odnawiania).

DANE GRAFICZNE (display data, graphic data) — 1) dowolny zbiór danych przeznaczonych dla urządzenia graficznego, 2) ROZKAZY GRAFICZNE I WSPÓŁRZĘDNE zawarte w BUFORZE OBRAZU, które bezpośrednio opisują OBRAZ FIZYCZNY.

DRUKARKA ELEKTROSTATYCZNA (electrostatic printer-plotter) — DRUKARKA GRAFICZNA, w której wykorzystuje się ładunki elektrostatyczne do tworzenia obrazu na elektroczułym nośniku.

DRUKARKA GRAFICZNA RASTROWA (raster printer-plotter) — GRAFICZNE URZĄDZENIE WYJŚCIOWE, które rejestruje (utrwala) obraz na nośniku metodą RASTROWEGO ROZWINIĘCIA OBRAZU.

DYSKRETYZATOR (digitizer) — czytnik rysunków służący do przetwarzania linii rysunków na postać cyfrową.

ELEMENT KREŚLĄCY OBRAZ (display writer) — element GRAFICZNEGO URZĄDZENIA WYJŚCIOWEGO służący do tworzenia widzialnego znaku (elementu graficznego) na POWIERZCHNI OBRAZOWANIA, np. pióro, promień lasera, promień elektronowy, rylec.

ELEMENT NIEWIDZIALNY (blanked element) — dane graficzne, których przesłanie do urządzenia graficznego nie wywołuje żandego efektu wizualnego lecz zmienia jedynie BIEŻĄCĄ POZYCJĘ ELEMENTU KREŚLĄCEGO OBRAZ.

ELEMENT OBRAZU (display element, display primitive, graphic primitive, output primitive) — element graficzny stanowiący jednorodną część obrazu fizycznego (np. punkt, odcinek prostej, znak alfanumeryczny itp., generowany na ogół sprzętowo).

ELEMENT WIDZIALNY (unblanked element) — DANE GRAFICZNE, których przesłanie do urządzenia graficznego wywołuje efekt wizualny (przeciwieństwo ELEMENTU NIEWIDZIALNEGO).

ELEMENT WYKRYWALNY (detectable element, selectable element) — ELEMENT OBRAZU mający ATRYBUTY, które pozwalają na WYKRYWANIE go za pomocą GRAFICZNEGO URZĄDZENIA WEJŚCIOWEGO (np. element może być wykrywany przez pióro świetlne).

GENERATOR KRZYWYCH (curve generator) — urządzenie lub układ elektroniczny, który generuje krzywą na podstawie definiujących ją danych.

GENERATOR OBRAZU (display generator) — urządzenie elektroniczne przetwarzające zawartość BUFORA OBRAZU na OBRAZ FIZYCZNY.

GENERATOR WEKTORÓW (vector generator) — urządzenie lub układ elektroniczny, który generuje odcinki linii prostych na podstawie zadanych punktów skrajnych bądź przyrostów wzdłuż osi układu. Wyróżnia się kilka rodzajów generatorów wektorów, m.in.: generator o proporcjonalnym czasie (proportional time vector generator), w którym czas kreślenia wektora jest w przybliżeniu proporcjonalny do długości tego wektora; generator o stałym czasie (constant time vector generator), w którym czas kreślenia nie zależy od długości wektora; generator przyrostowy (inoremental vector generator), który składa wektor z ciągu elementarnych odcinków linii siatki dyskretnej — tak, że wektory

[1]) W INFORMATYCE nr 9/1983 zamieściliśmy „Słowniczek terminów związanych z systemami graficznymi" autorstwa Jacka Owczarczyka — przyp. red.

[2]) Chmurzyński J. i in.: Grafoskopy w systemach komputerowych. WNT, Warszawa, 1978

[3]) Pierwszą wersję słownika opublikowano w Pracach Naukowo-Badawczych IMM, nr 1, 1983

[4]) Potto J.: Graphics Glossary. Computers and Graphics, 1978, vol. 3, No. 4, pp. 123—128

[5]) Dokument ISO/DIS 2382/13: Data processing vocabulary — Section 13, Computers graphics

nachylone mają postać zębatej piłki zamiast linii prostej.
GENERATOR ZNAKÓW (character generator) — urządzenie lub układ elektroniczny, który na podstawie podanego kodu generuje postać graficzną znaku. Wyróżnia się różne rdzaje generatorów znaków, m.in.: generator mozaikowy (dot matrix character generator), który tworzy znaki z kropek, i generator segmentowy (stroke character generator), który składa je z odcinków linii.
GŁOWICA KREŚLAKA (plotting head) — ruchoma część KREŚLAKA stanowiąca ELEMENT KREŚLĄCY OBRAZ.
GRAFICZNE URZĄDZENIE WEJŚCIOWE (graphic input device) — sprzęt umożliwiający wprowadzanie danych graficznych do komputera (p. MANIPULATOR RAMIENIOWY, PIÓRO ŚWIETLNE, RYSOWNICA, MYSZ MANIPULACYJNA, MANIPULATOR KULOWY).
GRAFICZNE URZĄDZENIE WYJŚCIOWE (display device) — urządzenie przeznaczone do przedstawiania obrazów na POWIERZCHNI OBRAZOWANIA. Zazwyczaj pojęcie to odnosi się do MONITORÓW GRAFICZNYCH, Z LAMPĄ KATODOWĄ, ale obejmuje również KREŚLAKI BĘBNOWE, KREŚLAKI STOŁOWE, MONITORY PLAZMOWE, REJESTRATORY MIKROFILMOWE oraz DRUKARKI GRAFICZNE.
GRAFIKA BIERNA (passive graphics) — używanie KONSOLI GRAFICZNEJ w TRYBIE BIERNYM. Zazwyczaj dotyczy wykorzystania takich urządzeń jak KREŚLAK, REJESTRATOR MIKROFILMOWY i DRUKARKA GRAFICZNA.
GRAFIKA INTERAKCYJNA (interactive graphics) — technika wykorzystania KONSOLI GRAFICZNEJ do prowadzenia komunikacji z komputerem w TRYBIE INTERAKCYJNYM.
GRAFIKA KOMPUTEROWA (computer graphics) — technika generowania i manipulowania obrazem za pomocą komputera. Odnosi się do obrazów, które zawierają linie, punkty, okręgi i inne figury geometryczne, a nie obejmuje techniki prezentowania danych alfanumerycznych.
GRAFIKA KONTUROWA (coded graphics, coordinate graphics, line graphics) — generowanie obrazów na podstawie ROZKAZÓW GRAFICZNYCH I WSPÓŁRZĘDNYCH techniką KOMPUTEROWEGO ROZWINIĘCIA OBRAZU.
GRAFIKA RASTROWA (OBRAZOWA) (image graphics, raster graphics) — generowanie obrazów na podstawie danych nie mających struktury wewnętrznej, takich jak faksymile, zdjęcia satelitarne lub obraz telewizyjny techniką RASTROWEGO ROZWINIĘCIA OBRAZU.
GRUBOŚĆ LINII (line weight) — względna szerokość lub intensywność linii wyświetlanej.
JEDNOSTKA RASTRA (raster unit) — odległość (mierzona w poziomie i w pionie) między dwoma sąsiednimi PUNKTAMI ADRESOWALNYMI na ekranie monitora.
JĘZYK GRAFICZNY (graphic language) — język umożliwiający programowanie urządzeń graficznych komputerów.
KADR OBRAZU (display frame) — wyróżniony OBRAZ EKSPONOWANY, jest to obraz powstający analogicznie do klatki w technice filmowej.
KINESKOP, LAMPA KATODOWA (cathode ray tube, CRT) — lampa elektronowa z ekranem pokrytym od wewnątrz substancją fosforyzującą (LUMINOFORM); (p. LAMPA PAMIĘTAJĄCA, LAMPA Z ODNAWIANIEM).
KINESKOP MASKOWY, CHROMATYCZNA MOZAIKOWA LAMPA KATODOWA (multicolor CRT) — kineskop, w którym powierzchnia ekranu pokryta jest punktowo umieszczonymi obok siebie LUMINOFORAMI o trzech podstawowych kolorach świecenia (zielonym, czerwonym, niebieskim), a strumienie elektronów pochodzące z trzech niezależnych dział elektronowych, przemieszczane przez wspólny układ odchylania, padając na ekran wytwarzają ten sam obraz w trzech podstawowych kolorach. Obserwator, wskutek zachodzącej w oku syntezy przestrzennej odbieranych zjawisk świetlnych, odbiera jeden obraz o różnych odcieniach barw.
KINESKOP PENETRACYJNY, CHROMATYCZNA PENETRACYJNA LAMPA KATODOWA (penetration CRT) — kineskop, którego ekran jest pokryty kilkoma warstwami LUMINOFORU o różnych kolorach świecenia, a każda warstwa może być selektywnie pobudzona przez strumień elektronowy o określonej energii. Zmianę koloru świecenia uzyskuje się przez zmiany energii strumienia elektronowego.
KLAWIATURA FUNKCYJNA (function keyboard) — urządzenie wejściowe KONSOLI GRAFICZNEJ złożone z pewnej liczby KLAWISZY FUNKCYJNYCH.
KLAWISZ FUNKCYJNY (function key, function button) — przycisk lub przełącznik służący do przesłania sygnału z KONSOLI GRAFICZNEJ do programu sterującego.

Jabłoński K.: Język programowania BCPL (1)

INFORMATYKA 1985, nr 2, s. 1

Pierwsza część charakterystyki języka BCPL. Omówiono genezę i ogólne cechy języka oraz podstawowe operacje, deklaracje i instrukcje.

Dziedzic M., Perycz K., Wiliński J.: Emulator E6RM-E jako przykład wykorzystania właściwości komputerów mikroprogramowanych

INFORMATYKA 1985, nr 2, s. 4

Charakterystyka rozwiązań przenoszenia oprogramowania z maszyn serii ODRA na komputery Jednolitego Systemu metodą emulacji z wykorzystaniem logiki mikroprogramowanej maszyn cyfrowych.

Kapuścik W.: Projektowanie sieciowych baz danych na przykładzie SZBD DMS-1100

INFORMATYKA 1985, nr 2, s. 6

Charakterystyka nowoczesnych metod projektowania logicznej oraz fizycznej struktury sieciowych baz danych, zapewniających poprawę efektywności ich eksploatacji.

Яблоньски К.: Язык программирования BCPL (1)

INFORMATYKA 1985, № 2, стр. 1

Первая часть характеристики языка BCPL. Обсуждаются происхождение, общие свойства языка основные операции, описания и операторы.

Дзедзиц М., Пэрыч К., Вилиньски И.: Эмулятор E6RM-E — пример использования свойств микропрограммных ЭВМ

INFORMATYKA 1985, № 2, стр. 4

Характеристика методов переноса программного обеспечения из ЭВМ ODRA на ЕС ЭВМ методом эмуляции с использованием микропрограммной логики вычислительных машин.

Капусьцик В.: Проектирование сетевых баз данных на примере SZBD DMS-1100

INFORMATYKA 1985, № 2, стр. 6

Характеристика современных методов проектирования логической и физической структуры сетевых баз данных, обеспечивающих повышение эффективности их эксплуатации.

Jabłoński K.: BCPL-programming language (1)

INFORMATYKA 1985, No. 2, p. 1

First part of the BCPL-language characteristics. Origin and general features of the language, as well as its basic operations, declarations and statements are discussed.

Dziedzic M., Perycz K., Wiliński J.: E6RM-E emulator as an example of utilization of microprogrammed computer features

INFORMATYKA 1985, No. 2, p. 4

Characteristics of solutions for software transfer from ODRA-computer series to Unified System computers using emulation with utilization of microprogrammed computer's logic.

Kapuścik W.: Network data base design on example of DMS-1100 Data Base Management System

INFORMATYKA 1985, No. 2, p. 6

Characteristics of modern methods for logical and physical network data base structure design, which secure effectiveness improvement of its operation.

Jabłoński K.: BCPL-Programmiersprache (1)

INFORMATYKA 1985, Nr. 2, S. 1

Erster Teil der BCPL-Sprachecharakteristik. Es wurden Ursprung und allgemeine Spracheeigenschaften, sowie grundlegende Operationen, Vereinbarungen und Anweisungen besprochen.

Dziedzic M., Perycz K., Wiliński J.: E6RM-E Emulator als Beispiel der Ausnützung von Eigenschaften der mikroprogrammierten Rechner

INFORMATYKA 1985, Nr. 2, S. 4

Eine Charakteristik von Lösungen der Softwareübertragung von ODRA-Rechnerreihe auf ESER-Rechner mit Verwendung der Emulationsmethode und Ausnützung der Logik von mikroprogrammierten Rechnern.

Kapuścik W.: Projektierung von Netzdatenbanken an Beispiel von DBS-1100 Datenbankverwaltungssystem

INFORMATYKA 1985, Nr. 2, S. 6

Eine Charakteristik der modernen Methoden für Projektierung der logischen und physischen Netzdatenbankstruktur, die eine Erhöhung ihrer Ausnutzungseffektivität sichern.

Terminologia

KLAWISZ ŚWIETLNY (virtual push button, light button) — znak graficzny lub napis na ekranie monitora będący elementem WYKAZU FUNKCYJNEGO, który może być WYKRYTY (wskazany) przez GRAFICZNE URZĄDZENIE WEJŚCIOWE (np. PIÓRO ŚWIETLNE) i działa jako KLAWISZ FUNKCYJNY.

KOD KADRU (display image) — DANE GRAFICZNE ZAKODOWANE W POSTACI ROZKAZÓW GRAFICZNYCH, tworzące jeden KADR OBRAZU.

KOMENDA GRAFICZNA (display command, display instruction) — polecenie wysyłane z komputera (w postaci sygnału) w celu sterowania urządzeniem graficznym (p. ROZKAZ GRAFICZNY).

KOMÓRKA OBRAZU (pixel, pel) — najmniejszy możliwy do wyświetlenia (niepodzielny) obszar obrazu cyfrowego na monitorze z rastrowym rozwinięciem obrazu, mający ATRYBUTY, które mogą się różnić od ATRYBUTÓW obszarów przylegających (p. ROZDZIELCZOŚĆ).

KOMPRESJA OBRAZU CYFROWEGO (scan compression) — wydzielenie i zakodowanie konturów obrazu cyfrowego.

KONSOLA GRAFICZNA (display console) — zestaw sprzętowy składający się co najmniej z jednego GRAFICZNEGO URZĄDZENIA WYJŚCIOWEGO i zazwyczaj jednego lub większej liczby urządzeń wejściowych, takich jak klawiatura alfanumeryczna, KLAWISZE FUNKCYJNE lub GRAFICZNE URZĄDZENIA WEJŚCIOWE.

KONTRAST (contrast) — zależność między intensywnością świecenia najjaśniejszych i najciemniejszych części obrazu, wyrażona zazwyczaj jako stosunek światła do cienia lub procentowo jako stosunek różnicy jaskrawości obrazu i jaskrawości tła do jaskrawości tła. Dotyczy tylko obrazów jednokolorowych (obrazów kolorowych dotyczy zupełnie inny rodzaj kontrastu.

KONTUROWE ROZWINIĘCIE (KREŚLENIE) OBRAZU (directed beam) — technika wykreślania ELEMENTÓW OBRAZU w kolejności podanej przez program komputerowy, gdzie ruch ELEMENTU KREŚLĄCEGO OBRAZ jest analogiczny do przesunięć pióra KREŚLAKA STOŁOWEGO.

KONTUROWE URZĄDZENIE GRAFICZNE (calligraphic display device, directod beam display device, directed beam device) — GRAFICZNE URZĄDZENIE WEJŚCIOWE wykreślające obraz metodą KONTUROWEGO ROZWINIĘCIA OBRAZU, np. MONITOR KONTUROWY, KREŚLAK.

KONWERSJA STRUKTURY OBRAZU (scan conversion) — sprzętowe lub programowe przekształcenie ciągu danych opisujących OBRAZ KONTUROWY na OBRAZ CYFROWY.

KOPIA KODOWA (soft copy) — obraz zapamiętany w postaci kodowej w celu przechowania, najczęściej na nośniku magnetycznym — pamięć taśmowa, dyskowa itp.

KOPIA TRWAŁA (hard copy) — obraz z monitora graficznego utrwalony na nośniku trwałym, jak papier, klisza fotograficzna, płytka szklana itp., możliwy do oglądania bez pośrednictwa GRAFICZNYCH URZĄDZEŃ WYJŚCIOWYCH.

KREŚLAK, PISAK (plotter) — KONTUROWE URZĄDZENIE GRAFICZNE, które utrwala obraz na nośniku trwałym, jak papier, film itp.

KREŚLAK ANALOGOWY (analog plotter) — KREŚLAK, którego pióro jest przesuwane za pomocą zmienianych w sposób ciągły (analogowy) sygnałów i może zostać ustawione na POWIERZCHNI OBRAZOWANIA w miejscu o dowolnych rozróżnialnych współrzędnych.

KREŚLAK BĘBNOWY (drum plotter) — KREŚLAK, który rysuje na materiale rejestrującym nawiniętym na bęben. Głowica (pióro KREŚLAKA porusza się równolegle do osi obrotu, a obrót bębna powoduje wykonanie rysunku wzdłuż drugiej osi układu.

KREŚLAK CYFROWY (digital plotter) — KREŚLAK, którego pióro może być przesuwane wyłącznie dyskretnymi krokami i umieszczane w określonych punktach POWIERZCHNI OBRAZOWANIA.

KREŚLAK STOŁOWY (PŁASKI) (flat bed plotter) — KREŚLAK, który rysuje obraz na płaskiej powierzchni (stole) ruchami wykonywanymi wyłącznie przez GŁOWICE KREŚLAKA.

KROK KREŚLAKA (plotter step size) — odległość mierzona w pionie lub poziomie pomiędzy sąsiednimi PUNKTAMI ADRESOWALNYMI KREŚLAKA CYFROWEGO.

WOJCIECH MOKRZYCKI
LENA WALKIEWICZ

■ Utworzona dwa lata temu brytyjska agencja rządowa ALVEY do popierania nowoczesnych przedsięwzięć rozwojowych w dziedzinie informatyki otrzymała na pięciolecie 1984/1985—1988/1989 budżet 200 mln funtów. Agencja opracowała cztery wielkie plany strategiczne: opracowywanie nowych układów mikroelektronicznych, wspomaganie narzędziowe prac programistycznych, tworzenie systemów banków wiedzy, zwiększanie komunikatywności w procesach współpracy ludzi i maszyn. Subsydia dla firm prywatnych udzielane są ex post do wysokości 50% nakładów badawczych na uprzednio zatwierdzone tematy, w ramach poszczególnych planów strategicznych — z horyzontem czasowym: 30 miesięcy, 5 i 10 lat. Ośrodki uniwersyteckie mogą ponadto korzystać z tradycyjnych form finansowania wąskich prac badawczych wchodzących w skład tematów ALVEY-owskich. Aby przyciągnąć do współpracy zdolnych informatyków i elektroników zatrudnionych w licznych małych przedsiębiorstwach — do których normalnymi kanałami subsydia rządowe praktycznie dotychczas nie trafiały, agencja ALVEY podjęła szeroką akcję propagandową w czasopismach fachowych i na wystawach przemysłowych. Oryginalna jest zwłaszcza forma powściągliwej agitacji za koordynowaniem dotychczas rozproszonych wysiłków firm odległych od czołówki światowej. (E)

*

■ Pomimo prognoz mówiących o spadku zamówień na produkty półprzewodnikowe, producenci działający w Stanach Zjednoczonych osiągają nadal rekordowe rezultaty. Oto bilans trzech wybranych firm.
— INTEL, wobec zysku 37 mln dol. w I półroczu 1983 roku osiągnęło I półroczu br. zysk 105 mld dol. Ogólny wzrost przychodów firmy wynosił, w zestawieniu kwartalnym, 58%. Firma odnotowuje piąty kwartał gwałtownego wzrostu popytu. Największe zapotrzebowanie zgłaszane jest na mikroprocesory. INTEL zwiększa ich produkcję, głównie — mikroprocesorów 8086 i 80286. Ze wzrostem produkcji rośnie zakres prac badawczych, koszty badawczo-rozwojowe wzrosły z 32 mln dol. w II kwartale 1983 do 44 mln dol. w analogicznym okresie ub. r.
— Oddział produkcji półprzewodnikowej MOTOROLI odnotowuje ostatnio największy wzrost zysków w zestawieniu z innymi działami tej firmy. Przy globalnym wzroście wartości towarów sprzedanych o 36%, wartość sprzedanych elementów półprzewodnikowych wzrosła o 49%. Zamówienia złożone w tym dziale aż o 100% przekraczają możliwości produkcyjne. W bieżącym kwartale MOTOROLA wprowadziła do sprzedaży mikroprocesor MC-68020 reklamowany jako mikroprocesor jednoukładowy o największych możliwościach funkcjonalnych w stosunku do innych urządzeń tego typu.
— Sprzedaż firmy SEEQ, działającej również w branży półprzewodnikowej, wzrosła ostatnio o 70% (do 8,8 mln dol.). Firma ma pakiet zamówień obejmujący sumę 45 mln dol. SEEQ ukierunkowuje produkcję na układy EEROM i EPROM o wysokiej sprawności. W bieżącym kwartale ukazały się serie próbne pamięci 256KB CMOS/EPROM oraz 64 KB CMOS EEROM z wbudowaną korekcją błędów. (W).

*

■ Firma RB ROBOT ze stanu Korolado (USA) oferuje „roboty domowe". Robot RB5X, produkowany przez tę firmę, reklamowany jest jako inteligentny, programowany sprzęt domowy. Urządzenie to może — dzięki czujnikom dotyku — swobodnie poruszać się w umeblowanym mieszkaniu. Gdy akumulatory RB5X są bliskie wyczerpania, robot sam odnajduje zasilacz i łączy się ze źródłem zasilania. Robot zaopatrzony jest w cały zestaw przystawek pozwalających wykonywać różne prace domowe. Połączenie RB5X z odkurzaczem pozwala np. na sprzątanie mieszkania po wyjściu właścicieli. Montując ramię z częścią chwytającą, można uczynić z robota posłańca przenoszącego lekkie przedmioty (do 0,35 kg). Instalacja czujnika dymu oraz gaśnicy daje natomiast „domowego strażaka". Firma RB ROBOT oferuje egzemplarze RB5X zaopatrzone dodatkowo w urządzenie rozpoznawania mowy — roboty te mogą wykonywać słowne polecenia oraz komunikować o ich wykonaniu. Dostarczane wraz z RB5X języki sterowania RCL (Robot Central Language) oraz system Sarry — umożliwiają szybkie i łatwe programowanie robota. (W).

*

■ Maszyny rodziny LAMBDA LMI, zwane również maszynami lispowymi, otrzymały ostatnio system trójwymiarowej kolorowej garfiki czasu rzeczywistego z podsystemem wysokiej rozdzielczości Silicon Graphic Iris. System ten rozszerza możliwości przetwarzania symbolicznego maszyn LAMBDA, stwarza bowiem idealne warunki do rozwiązywania zagadnień symulacji złożonej, które wymagają oprogramowania w językach sztucznej inteligencji oraz manipulacji (w czasie rzeczywistym) trójwymiarowymi obrazami. Podsystem Iris ma pamięć obrazu o rozmiarze 1024×1024 komórek, wyświetlaną na ekranie 19-calowego monitora kolorowego. Komórki pamięci obrazu są 12-bitowe, z możliwością rozszerzenia do 24 bitów, co pozwala zrealizować 16 mln kolorów. Iris, składający się ze 150 podprogramów, pozwala na animację komputerową z prędkością przekraczającą 65 tys. trójwymiarowych przekształceń na sekundę. Tak dużą szybkość umożliwia dziesięć tzw. maszyn geometrycznych realizujących równocześnie wszelkie przekształcenia geometryczne. Cena podsystemu Iris wynosi 40 tys. dol. Wieloprocesorowe maszyny LAMBDA uważane są za komputery kolejnej generacji. Maszyny te mogą mieć aż cztery szybkie procesory, a używane są do rozwiązywania zagadnień użytkowych za pomocą metod sztucznej inteligencji (W).

W skrócie • W skrócie • W skrócie • W skrócie • W skrócie • W skrócie

Zasady prenumeraty

Prenumeratorzy zbiorowi — jednostki gospodarki uspołecznionej, instytucje i organizacje społeczne zamawiają prenumeratę dokonując wpłaty na blankiecie „polecenie przelewu" rozszerzonym dla potrzeb Wydawnictwa o część dotyczącą zamówienia.
Blankiety te będą dostarczone przez Zakład Kolportażu.

Prenumeratorzy indywidualni — osoby fizyczne zamawiają prenumeratę dokonując wpłaty w UPT lub NBP na blankiecie Wydawnictwa lub blankiecie NBP. Na odwrocie wszystkich odcinków blankietu należy wpisać tytuł czasopisma, okres prenumeraty, liczbę zamawianych egzemplarzy oraz wartość wpłaty.
Wpłacać należy na konto NBP III O/M Warszawa 1036-7490-139-11.

Prenumerata ulgowa — przysługuje wyłącznie osobom fizycznym — członkom SNT, studentom i uczniom szkół zawodowych. Warunkiem prenumeraty ulgowej jest poświadczenie blankietu wpłaty (przed jej dokonaniem) na wszystkich odcinkach pieczęcią Koła SNT, wyższej uczelni lub szkoły.

Sposób zamawiania prenumeraty taki sam jak dla prenumeraty indywidualnej.

Prenumerata ze zleceniem wysyłki za granicę — zamawia się tak jak prenumeratę indywidualną. Dodatkowo należy podać na blankiecie wpłaty nazwisko i dokładny adres odbiorcy.
Cena prenumeraty ze zleceniem wysiłki za granicę jest dwukrotnie wyższa.

Przedpłaty na prenumeratę przyjmowane są w terminach:
— do 10 listopada na I kwartał, I półrocze i cały rok następny,
— do 28 lutego na II, III, IV kwartał i II półrocze,
— do 31 maja na III, IV kwartał i II półrocze,
— do 31 sierpnia na IV kwartał.

U w a g a

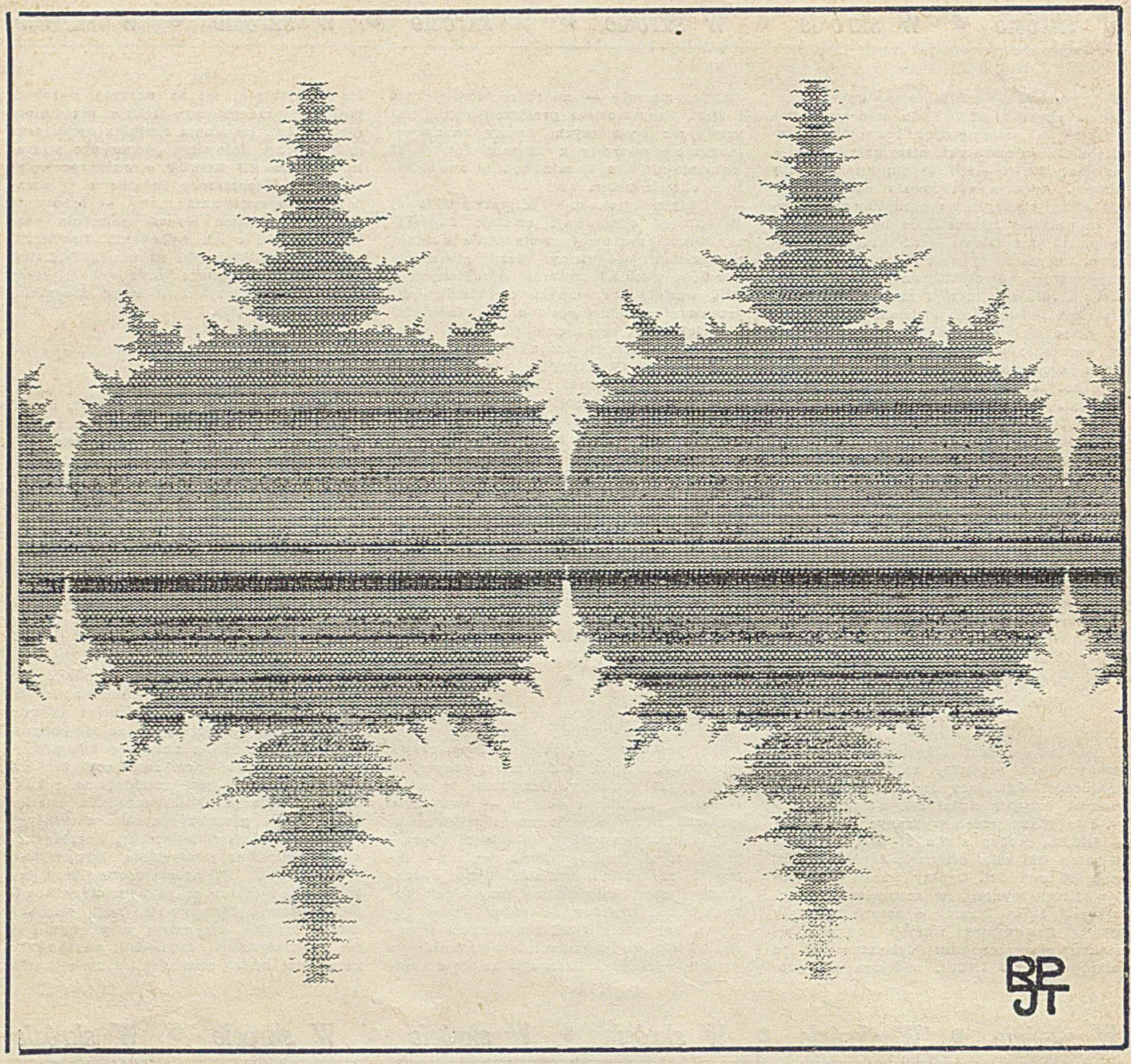
Przedpłata na dwumiesięczniki przyjmowana jest na okresy półroczne lub roczne.

Informacji o prenumeracie udziela — Zakład Kolportażu Wydawnictwa NOT-SIGMA, ul. Bartycka 20, 00-716 Warszawa, lub skr. poczt. 1004, 00-950 Warszawa, tel. 40-00-21 w. 249, 293, 297, 299 oraz 40-35-89.

Egzemplarze archiwalne czasopism — można nabyć za gotówkę w Klubie Pracy Technicznej w Warszawie ul. Mazowiecka 12, tel. 27-43-65 oraz w Dziale Handlowym Wydawnictwa ul. Bartycka 20, skr. poczt. 1004, 00-950 Warszawa, na rachunek dla instytucji, lub za zaliczeniem pocztowym dla osób fizycznych.

Cena prenumeraty INFORMATYKI (w złotych)

kwartalna		półroczna		roczna	
normalna	ulgowa	normalna	ulgowa	normalna	ulgowa
300,—	105,—	600,—	210,—	1200,—	420,—



FRAKTALE

Grafika przedstawia jeden z przykładów tzw. fraktali, czyli tworów geometrycznych o wymiarach niecałkowitych. Teoria fraktali została opracowana przez Benoit Mandelbrota, matematyka amerykańskiego (pracuje w IBM) pochodzenia polskiego (ur. w roku 1924 w Warszawie). Wszystkim, którzy lubią piękno w matematyce polecamy rewelacyjną książkę Mandelbrota „The Fractal Geometry of Nature" (W. H. Freeman and Co., San Francisco 1982, ISBN 0-7167-1186-9), w której przedstawiono parę setek różnych fraktali, m.in. piękne „drzewa", które przypominają śnieżynki (wzrost kryształków lodu od komórki elementarnej do makrokryształów jest przykładem tworu samopodobnego, tzn. każda część większej całości jest podobna do mniejszych jednostek strukturalnych). Nasz fraktal został narysowany przez kolejne interacje zespolonej funkcji sinus — jeżeli sekwencja sin(sin(sin...(z)...) jest zbieżna do zera, to zaczerniamy punkt z; gdy natomiast jest rozbieżna, to taki punkt pozostawiamy jasny. Dlaczego taka konstrukcja rzeczywiście tworzy fraktal można przeczytać w Mathematical Inteligencer vol. 6, No. 2 str. 11 (1984). Dodajmy, że ciemniejsze pasy na rysunku są li tylko wynikiem złego działania naszego plotera — tym razem grafika była rysowana ponad dwadzieścia godzin i zmiany temperatury w pomieszczeniu komputera (niestety bez klimatyzacji!) powodowały nierównomierny wypływ tuszu. Uznaliśmy jednak, że taką losową „korekcję" przyrody można przyjąć.

RAFAŁ PIETRAK
JAKUB TATARKIEWICZ